✦ vibecodingthailand.com

FREE

POCKET BOOK · 2026

Claude + Obsidian คลังความรู้ส่วนตัวที่ AI ใช้งานได้จริง

วิธีจัดโน้ต งาน ไอเดีย และแหล่งอ้างอิงให้กลายเป็น LLM Wiki แบบอ่านง่าย ไม่ต้องเขียนโค้ด

- ✓ เก็บความรู้กระจัดกระจายให้กลายเป็น Wiki ที่ค้นได้
- ✓ ให้ Claude อ่าน context งานของเราได้ดีขึ้น
- ✓ เริ่มได้ในวันแรก ไม่ต้องเขียนโค้ดสักบรรทัด

7 บท · 7 APPENDIX · 39 ภาพประกอบ

FIRST EDITION · 2026

สารบัญ

ทำไม Prompt อย่างเดียวไม่พอ

หมิวเป็น marketing coordinator

งานของหมิวไม่ได้ซับซ้อนแบบนักพัฒนา software

แต่ข้อมูลในชีวิตทำงานของหมิวกระจัดกระจายมาก

บางอย่างอยู่ใน Google Docs

บางอย่างอยู่ใน LINE

บางอย่างอยู่ใน email

บางอย่างอยู่ใน meeting note

บางอย่างอยู่ใน PDF

บางอย่างอยู่ในหน้าเว็บคู่แข่ง

บางอย่างอยู่ใน chat เก่ากับ Claude

และบางอย่างอยู่ในหัวหมิวเอง

เวลาต้องทำงานจริง เช่น ร่าง FAQ หน้าเว็บคอร์ส AI หมิวไม่ได้ขาดข้อมูล

หมิวมีข้อมูลเยอะเกินไปด้วยซ้ำ

ปัญหาคือ:

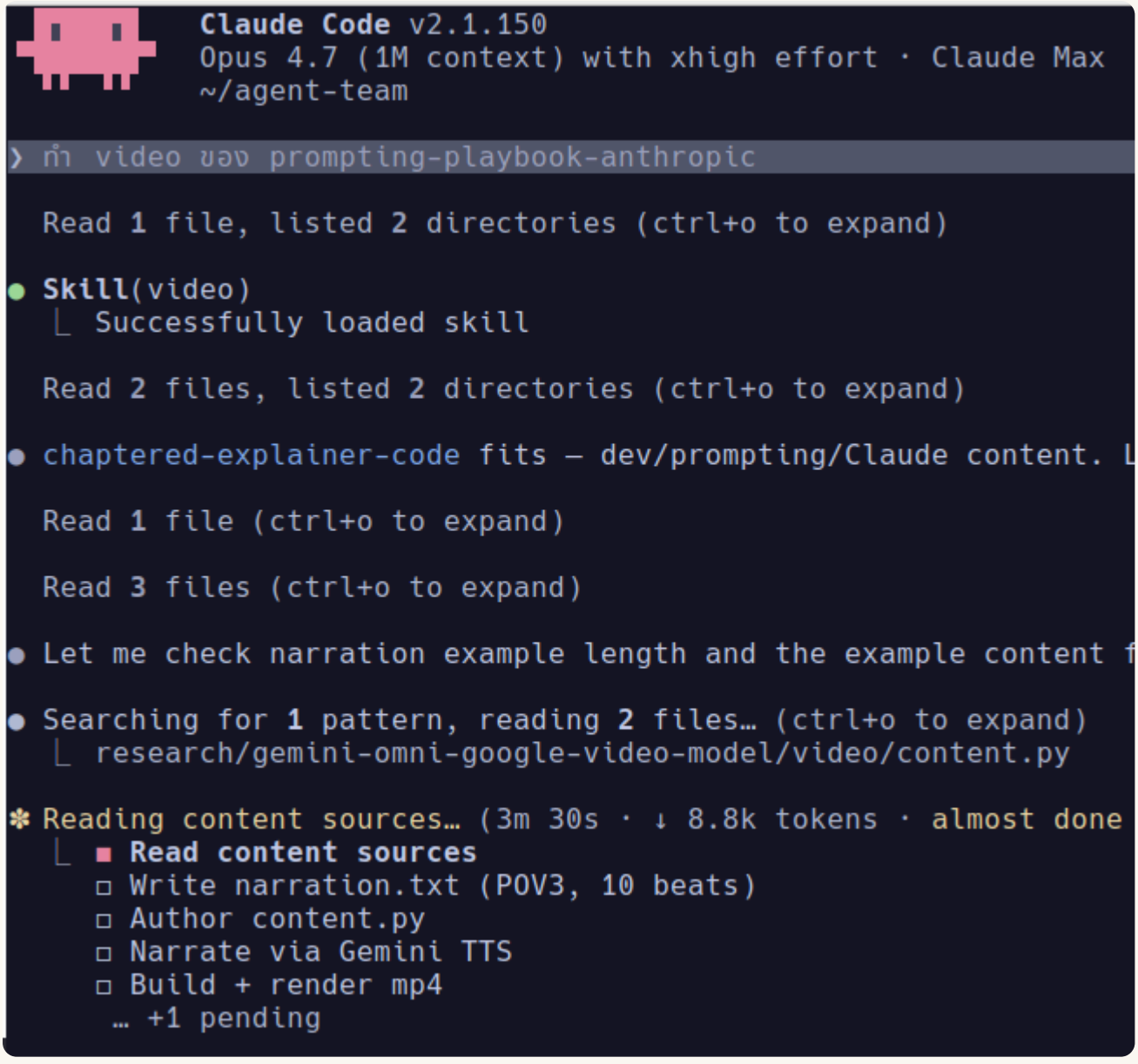
> หาไม่เจอ
> จำไม่ได้ว่าอยู่ที่ไหน
> ไม่รู้ว่าอันไหนเป็นข้อมูลจริง
> ไม่รู้ว่าอันไหนเป็นไอเดีย
> ไม่รู้ว่าอันไหนทีมตัดสินใจแล้ว
> และ `Claude` ก็ไม่รู้บริบทงานของหมิวพอ

นี่คือปัญหาของคนทำงานยุค AI

เราไม่ได้เริ่มจากหน้ากระดาษว่าง

เราเริ่มจากข้อมูลกระจัดกระจาย

AI เก่งขึ้น แต่ความรู้ของเรายังรกเหมือนเดิม



ตัวอย่าง Claude กำลังช่วยอ่านและจัดงานจากคำสั่งของเรา

ภาพ: ตัวอย่าง Claude กำลังช่วยอ่านและจัดงานจากคำสั่งของเรา

หลายคนเริ่มใช้ Claude ด้วยวิธีง่าย ๆ:

> ช่วยสรุปให้หน่อย
> ช่วยเขียนให้หน่อย
> ช่วยคิดหัวข้อให้หน่อย
> ช่วยทำ `email` ให้หน่อย

วิธีนี้ดี

และใช้ได้จริง

แต่พอใช้ไปสักพัก จะเจอปัญหาเดิมซ้ำ ๆ

Claude ช่วยได้ดีในหนึ่ง chat

แต่พอวันต่อมา คุณต้องเล่า context ใหม่

พอเปลี่ยนเรื่อง คุณต้อง upload file ใหม่

พอถามเรื่องเดิม คุณต้องขุดเอกสารเดิมมาวางใหม่

พอ chat ยาวเกินไป คำตอบเริ่มหลุดจากบริบท

สุดท้ายคุณไม่ได้ทำงานกับ “ความรู้ของตัวเอง”

คุณทำงานกับ “เศษ context ที่เอาไปให้ AI ทีละครั้ง”

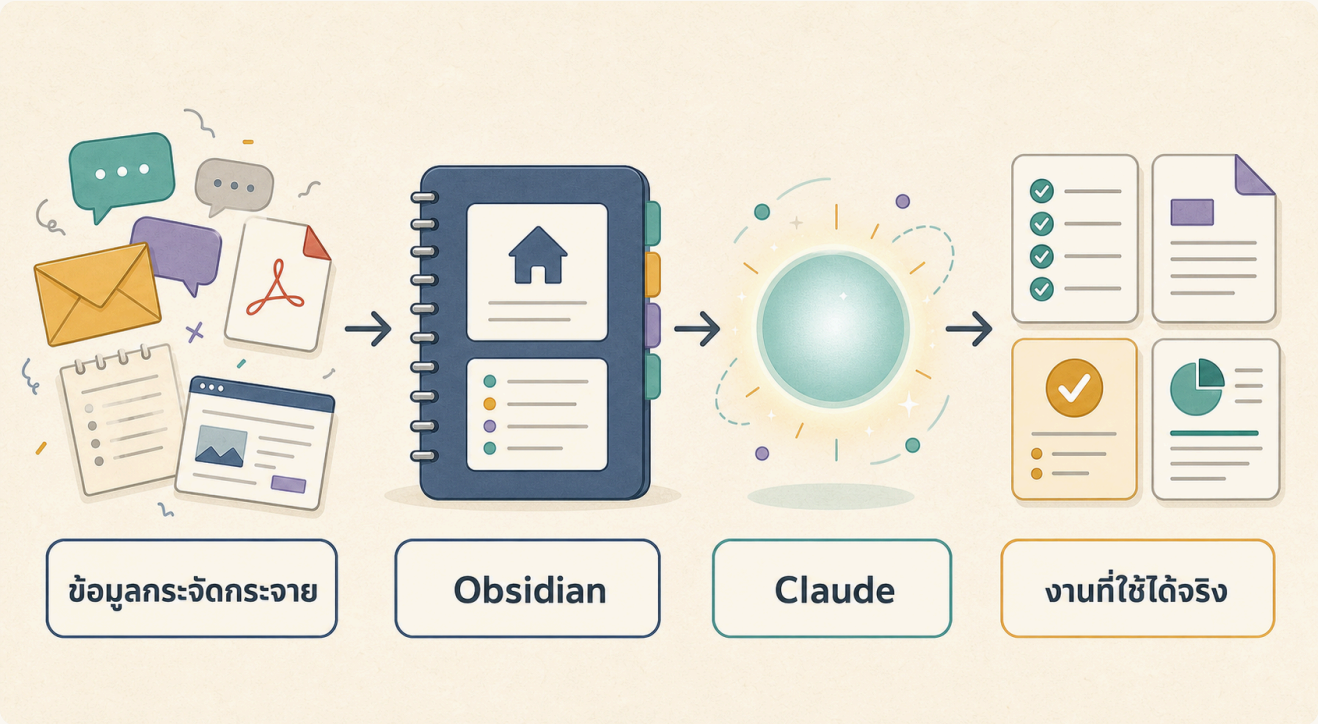
Prompt จึงไม่พอ

ไม่ใช่เพราะ prompt ไม่สำคัญ

แต่เพราะ prompt เป็นแค่คำถาม

ถ้าคลังความรู้ต้นทางยังรก คำถามที่ดีแค่ไหนก็ช่วยได้จำกัด

ปัญหาที่หนังสือเล่มนี้จะแก้



เมื่อข้อมูลกระจัดกระจายถูกจัดเข้าบ้านของความรู้ Claude จะช่วยอ่าน คิด และร่างงานได้ดีขึ้น

ภาพ: เมื่อข้อมูลกระจัดกระจายถูกจัดเข้าบ้านของความรู้ Claude จะช่วยอ่าน คิด และร่างงานได้ดีขึ้น

หนังสือเล่มนี้ไม่ได้เริ่มจากคำถามว่า:

> จะ prompt Claude ให้เทพยังไง

แต่เริ่มจากคำถามว่า:

> จะจัดความรู้ของเราให้ Claude อ่านแล้วช่วยงานได้ดีขึ้นยังไง

ต่างกันมาก

ถ้าความรู้ของคุณกระจัดกระจาย Claude ต้องเดาเยอะ

ถ้าความรู้ของคุณมี source, link, summary, decision และ index ชัด Claude จะช่วยได้ตรงขึ้น

สิ่งที่เราจะทำในเล่มนี้คือสร้างระบบเล็ก ๆ ที่เรียกว่า LLM Wiki

คำว่า LLM Wiki ในเล่มนี้หมายถึง:

> คลังความรู้ที่เขียนให้ทั้งคนและ AI อ่านรู้เรื่อง มีที่มา มี link มี index และมีรูปแบบ note ที่ชัด

ไม่ต้องรู้ code

ไม่ต้องทำระบบ technical

ไม่ต้องเริ่มจาก plugin

เริ่มจากการจัด note ให้ดีพอที่คุณอ่านรู้เรื่อง และ Claude อ่านแล้วเข้าใจบริบทงานของคุณ

คำอังกฤษที่ใช้ในเล่มนี้

เล่มนี้ตั้งใจใช้ภาษาไทยแบบคนทำงานจริง คือไม่แปลทุกคำให้ฝืน

คำต่อไปนี้จะเจอบ่อย ให้เข้าใจประมาณนี้พอ:

```
Source   = ต้นทางข้อมูล เช่น meeting note, email, PDF, หน้าเว็บ
Prompt   = คำสั่งหรือโจทย์ที่ส่งให้ Claude
Context  = สิ่งที่ให้ Claude อ่านเพื่อเข้าใจงานก่อนตอบ
Draft    = งานร่าง ยังไม่ใช่ final
Review   = การตรวจทาน
Project  = งานหรือโปรเจกต์ที่ต้องทำต่อ
Concept  = แนวคิดที่ใช้ซ้ำได้
Decision = การตัดสินใจที่ตกลงแล้ว
```

ไม่ต้องจำศัพท์ให้ครบก่อนอ่าน

อ่านไปแล้วจะเห็นตัวอย่างซ้ำ ๆ จนเข้าใจเอง

เครื่องมือหลักมี 2 ตัว

เล่มนี้ใช้เครื่องมือหลักสองตัว

1. Obsidian = บ้านของความรู้

Obsidian คือที่เก็บ note ของคุณ

ให้คิดง่าย ๆ ว่ามันคือบ้านของไฟล์ความรู้

แต่ละ note เป็นไฟล์ข้อความธรรมดา

คุณสร้าง folder ได้

คุณ link note เข้าหากันได้

คุณทำ Home หรือ Index ได้

คุณเก็บ source ได้

สิ่งสำคัญคือความรู้ไม่ได้ลอยอยู่ใน chat เดียว

มันอยู่ในที่ที่คุณกลับมาเปิด อ่าน แก้ และจัดต่อได้

ในเล่มนี้ เราจะใช้ Obsidian แบบง่ายมาก

ไม่เน้น plugin

ไม่เน้น graph สวย

ไม่เน้นระบบซับซ้อน

เน้นให้คนทำงานทั่วไปใช้ได้จริง

2. Claude = ผู้ช่วยอ่านและจัดบ้าน

Claude คือผู้ช่วยที่ช่วยอ่าน สรุป แยกประเด็น ถามกลับ และร่างงาน

แต่ Claude ไม่ใช่ที่เก็บความรู้หลัก

และไม่ใช่เจ้าของความจริง

Claude ช่วยได้มากเมื่อคุณให้ context ดี

เช่น:

- source note ที่มีที่มา
- concept note ที่อธิบายเรื่องสำคัญ
- project note ที่บอกว่างานนี้กำลังทำอะไร
- decision note ที่บอกว่าทีมตกลงอะไรไปแล้ว

- Home หรือ index ที่บอกว่าเรื่องสำคัญอยู่ตรงไหน

พูดสั้น ๆ:

> Obsidian = บ้านของความรู้
> Claude = ผู้ช่วยอ่านและจัดบ้าน
> คุณ = เจ้าของบ้านและ editor สุดท้าย

นี่คือแกนของหนังสือทั้งเล่ม

แล้ว LLM Wiki ต่างจากการจดโน้ตธรรมดายังไง

โน้ตธรรมดามักเป็นแบบนี้:

> ประชุมทีม sales 25 พ.ค.
> ลูกค้าถามเรื่อง AI training
> ต้องทำ FAQ
> อาจทำ webinar

อ่านตอนจดใหม่ ๆ พอเข้าใจ

แต่ผ่านไปสองสัปดาห์ คุณอาจลืมว่า:

- meeting นี้เกี่ยวกับ project ไหน
- อะไรเป็นคำถามจริงจากลูกค้า
- อะไรเป็นไอเดียของทีม
- อะไรตัดสินใจแล้ว
- deadline คือเมื่อไหร่
- note นี้ควรไป link กับอะไร

LLM Wiki จะทำให้ข้อมูลเดียวกันชัดขึ้น เช่น:

```
Source note = meeting นี้มาจากไหน
Concept note = AI training สำหรับคนไม่รู้โค้ดคืออะไร
Project note = FAQ หน้าเว็บคอร์ส AI ต้องทำอะไรต่อ
Decision note = ทีมตัดสินใจอะไรแล้ว
Home/Index = เริ่มอ่านเรื่องนี้จากตรงไหน
```

ไม่ได้ยาวขึ้นเพื่อให้ดูเป็นระบบ

แต่ชัดขึ้นเพื่อให้ใช้ต่อได้

และเมื่อ Claude อ่าน note แบบนี้ Claude จะไม่ต้องเดาเยอะเท่าเดิม

หนังสือเล่มนี้ไม่ใช่อะไร

เพื่อให้คาดหวังตรงกัน เล่มนี้ไม่ใช่:

- คู่มือ Obsidian แบบครบทุกปุ่ม
- คู่มือ Claude แบบครบทุกฟีเจอร์
- หนังสือสอนเขียน code
- หนังสือทำ plugin
- หนังสือทำระบบ RAG หรือฐานข้อมูล vector
- หนังสือ productivity ที่ให้จัดชีวิตทั้งชีวิต
- หนังสือทฤษฎี PKM ยาว ๆ

เล่มนี้ตั้งใจเป็น pocket book สั้น ๆ สำหรับคนทำงานทั่วไป

อ่านแล้วควรลงมือได้

ถ้าคุณเป็น developer คุณอาจรู้สึกว่าเล่มนี้เรียบง่ายมาก

นั่นคือความตั้งใจ

เพราะผู้อ่านหลักคือคนที่อยากใช้ AI กับงานเอกสาร งานประชุม งาน content งาน sales งาน marketing งาน HR งาน admin หรืองาน project ทั่วไป

ไม่ใช่คนที่อยากสร้าง software system

อ่านจบแล้วคุณควรทำอะไรได้

อ่านจบเล่มนี้ คุณควรทำได้ 7 อย่าง

> 1. เข้าใจว่า Obsidian เป็นบ้านของความรู้ ไม่ใช่แค่แอปจดโน้ต
> 2. เข้าใจว่า Claude เป็นผู้ช่วย ไม่ใช่เจ้าของความจริง
> 3. แยก note เป็น Source / Concept / Project / Decision / Index ได้
> 4. ตั้ง vault แรกแบบง่ายได้ใน 30 นาที
> 5. เอา source หนึ่งชิ้นเข้า workflow แล้วแปลงเป็น Wiki ที่ใช้ต่อได้
> 6. ทำ Weekly Review เพื่อไม่ให้ Wiki รก
> 7. ใช้ Claude กับงานจริงแบบระวัง source และข้อมูลลับ

ถ้าทำได้เท่านี้ คุณไม่จำเป็นต้องมีระบบสมบูรณ์แบบ

คุณก็เริ่มได้แล้ว

โครงสร้างของเล่ม

เล่มนี้เดินแบบสั้นและเป็นขั้น

บทที่ 1: Obsidian คือบ้านของความรู้

เริ่มจากเข้าใจ Obsidian แบบไม่ technical

Vault คืออะไร

Note คืออะไร

Link, backlink, tag, property, Home, graph, canvas, web clipper ใช้ทำอะไรในภาษาคนทำงาน

บทที่ 2: Claude คือผู้ช่วยอ่านและจัดระบบ

วางบทบาทของ Claude ให้ถูก

Claude ช่วยอ่าน source, แตก note, เชื่อมโยง, ถามกลับ, ช่วยร่างงาน

แต่ Claude ไม่ใช่ความจำถาวร และไม่ใช่เจ้าของความจริง

บทที่ 3: LLM Wiki คืออะไร

อธิบาย pattern หลักของ Wiki:

Source note, Concept note, Project note, Decision note และ Index note

พร้อมตัวอย่างจาก meeting note จริง ๆ

บทที่ 4: ตั้งค่า Vault แรกใน 30 นาที

ลงมือสร้าง vault, folder, Home, Log และ template ง่าย ๆ

ไม่เริ่มจาก plugin

เริ่มจากบ้านเล็ก ๆ ที่ใช้งานได้

บทที่ 5: จาก source หนึ่งชิ้นเป็น Wiki ที่ใช้ต่อได้

บทหลักของ workflow

Capture → Clean → Ask Claude → Split → Link → Save → Review

ใช้กับ meeting note, email, article, webpage หรือ chat เก่าได้

บทที่ 6: ใช้ Claude ดูแล Wiki ไม่ให้รก

เมื่อ Wiki โตขึ้น ต้องมี Weekly Review

ล้าง inbox, หา note ซ้ำ, เช็ก project, ปรับ Home, เขียน Log

ใช้ Claude ช่วยตรวจบ้าน แต่คุณยังเป็นคนตัดสินใจ

บทที่ 7: ก่อนเอาไปใช้จริง: 5 กติกากันพลาด

ปิดเล่มด้วยกติกาสั้น ๆ:

อย่าส่งข้อมูลลับถ้าไม่จำเป็น

อย่าเชื่อ Claude ถ้าไม่มี source

แยก fact / inference / idea / decision

อย่าให้ระบบใหญ่กว่างาน

และทบทวนเป็นรอบ

เล่มนี้ใช้กับงานอะไรได้บ้าง

ตัวอย่างงานที่เหมาะกับวิธีในเล่มนี้:

- สรุป meeting แล้วทำงานต่อ
- เก็บคำถามลูกค้าเพื่อทำ FAQ
- รวบรวม research สำหรับเขียนบทความ
- จัด knowledge base ของทีมเล็ก ๆ
- เตรียม brief ให้ Claude ช่วยร่าง content
- เก็บ decision ของ project ไม่ให้หาย
- ทำคลัง prompt และตัวอย่างงานที่ใช้ซ้ำได้

งานเหล่านี้ไม่ได้ต้องการระบบใหญ่

ต้องการแค่ที่เก็บความรู้ที่หาเจอ มีที่มา และส่งต่อให้ AI อ่านได้ง่าย

วิธีอ่านเล่มนี้

ถ้าคุณมีเวลา อ่านเรียงบทจะดีที่สุด

เพราะแต่ละบทต่อกัน:

> บ้าน → ผู้ช่วย → รูปแบบ `note` → ตั้งบ้าน → `workflow` → ดูแลบ้าน → กันพลาด

แต่ถ้าคุณอยากลงมือเร็ว ให้ทำแบบนี้:

1. อ่านบทที่ 1 แบบผ่าน ๆ เพื่อเข้าใจ Obsidian
2. อ่านบทที่ 4 แล้วสร้าง vault
3. อ่านบทที่ 5 แล้วเอา source จริงหนึ่งชิ้นเข้าระบบ
4. กลับมาอ่านบทที่ 2, 3, 6, 7 เพื่อทำให้ระบบดีขึ้น

อย่ารอให้เข้าใจทั้งหมดก่อนเริ่ม

ระบบนี้เข้าใจดีที่สุดตอนลงมือทำกับงานจริง

ตัวอย่างที่จะใช้ตลอดเล่ม

เพื่อให้เห็นภาพ เราจะใช้ตัวอย่าง “หมิว” เป็นหลัก

หมิวกำลังช่วยทีมขายและทีม marketing ทำ FAQ หน้าเว็บสำหรับคอร์ส AI training

source แรกของหมิวคือ meeting note สั้น ๆ:

```
ประชุมทีม sales 25 พ.ค.
ลูกค้าถามบ่อยว่า AI training ต้องรู้โค้ดไหม
หลายคนกลัวทีมใช้ไม่เป็น
ทีม sales อยากได้ FAQ หน้าเว็บ
ควรมี webinar สำหรับผู้บริหารที่ไม่รู้ technical
ต้องส่ง draft FAQ ภายในศุกร์นี้
```

note นี้ดูธรรมดามาก

แต่ในเล่มนี้ เราจะใช้มันให้เห็นว่า source หนึ่งชิ้นสามารถกลายเป็น:

- source note
- concept note
- project note
- idea หรือ decision ที่แยกชัด
- link ใน Home
- item ใน Log
- prompt ที่ Claude ใช้ช่วยทำงานต่อ

นี่คือแกนการฝึกของเล่ม

คุณไม่จำเป็นต้องใช้ตัวอย่างเดียวกับหมิว

ให้แทนด้วย source ล่าสุดของคุณเอง

meeting note ล่าสุด

email ล่าสุด

brief ล่าสุด

PDF ล่าสุด

หรือ chat เก่ากับ Claude ที่อยากเก็บไว้

เริ่มจากเล็กที่สุด

ถ้าหนังสือเล่มนี้มีคำแนะนำเดียว คือ:

> เริ่มจาก source หนึ่งชิ้น

อย่าเริ่มจากการย้ายเอกสารทั้งหมดเข้า Obsidian

อย่าเริ่มจากการติด plugin 20 ตัว

อย่าเริ่มจากการออกแบบระบบสมบูรณ์แบบ

เริ่มจาก note เดียว

ทำให้มันมีที่มา

ทำให้มันอ่านรู้เรื่อง

ให้ Claude ช่วยแยก fact / inference / idea

แตกเป็น note เท่าที่จำเป็น

link กลับไปมา

แล้วสัปดาห์หน้าค่อยกลับมา review

แค่นี้คือจุดเริ่มต้นของ LLM Wiki แล้ว

ถ้าคุณทำซ้ำได้ ความรู้ของคุณจะค่อย ๆ เปลี่ยนจากกองข้อมูลกระจัดกระจาย เป็นระบบเล็ก ๆ ที่ช่วยให้คุณทำงานกับ AI ได้ดีขึ้นจริง

พร้อมแล้ว เริ่มจากบ้านก่อน

ก่อนจะให้ Claude ช่วยอ่านบ้าน

เราต้องมีบ้านก่อน

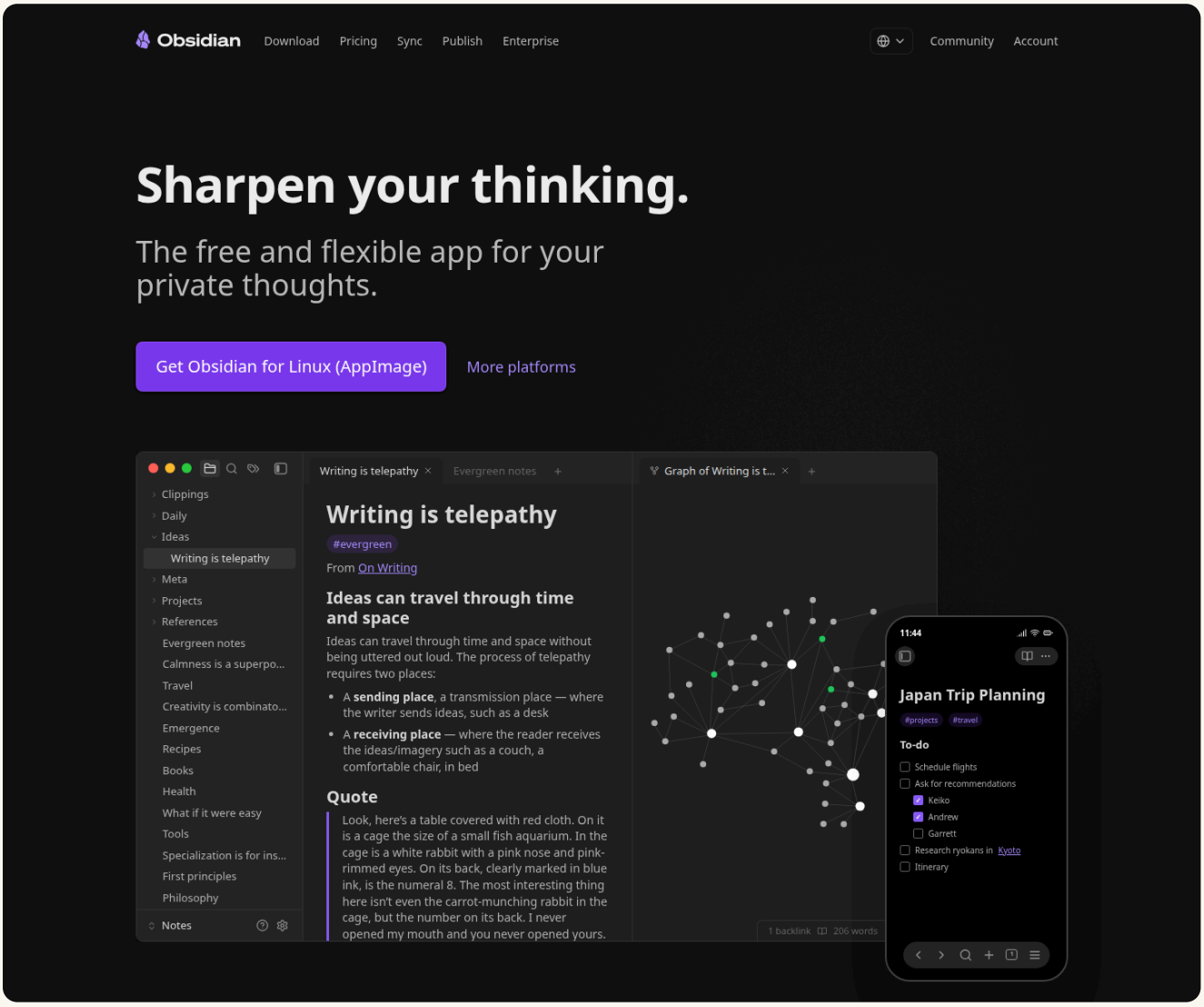
บทแรกจึงเริ่มที่ Obsidian

ไม่ใช่ในฐานะแอปจดโน้ตอีกตัว

แต่ในฐานะบ้านของความรู้ที่คุณคุมเอง

บทที่ 1

Obsidian คือบ้านของความรู้ ไม่ใช่แค่แอปจดโน้ต



หน้าเว็บ Obsidian อย่างเป็นทางการ: app สำหรับจดและเชื่อมโยงความคิด

ภาพ: หน้าเว็บ Obsidian อย่างเป็นทางการ: app สำหรับจดและเชื่อมโยงความคิด

ถ้าคุณเคยมีอาการแบบนี้ แปลว่าบทนี้เขียนมาเพื่อคุณ:

- จำไม่ได้ว่าไฟล์สำคัญอยู่ใน Google Drive, LINE, email หรือ chat กับ Claude
- เคยให้ Claude ช่วยสรุปงานไว้ดีมาก แต่หาไม่เจอแล้ว

- มีลิงก์ดี ๆ เต็ม bookmark แต่ไม่เคยกลับไปอ่าน
- จด meeting note ไว้หลายที่ แต่เอามารวมเป็นภาพเดียวไม่ได้
- เริ่มใช้ AI แล้วรู้สึกว่า “ถ้า AI รู้ context งานเรามากกว่านี้ คงช่วยได้ดีกว่านี้”

ปัญหานี้ไม่ได้แก้ด้วย prompt ที่ยาวขึ้นอย่างเดียว

สิ่งที่ต้องมีคือ “บ้านของความรู้”

สำหรับเล่มนี้ บ้านหลังนั้นคือ **Obsidian**

ไม่ใช่เพราะ Obsidian สวยที่สุด หรือมี feature เยอะที่สุด แต่เพราะมันเหมาะกับงานหนึ่งมาก: เก็บความรู้เป็นไฟล์ธรรมดาที่คนอ่านได้ AI อ่านได้ และเชื่อมโยงกันได้

บทนี้จะไม่สอนทุกปุ่มของ Obsidian เราจะเอาแค่ของที่ต้องรู้จริง เพื่อให้คุณเริ่มสร้าง LLM Wiki ของตัวเองได้

---

1. Obsidian คืออะไรแบบภาษาคนทำงาน

ให้คิดว่า Obsidian คือโฟลเดอร์งานที่ฉลาดขึ้น

ปกติคุณอาจมีโฟลเดอร์แบบนี้:

> งานบริษัท
> `meeting notes`
> ลูกค้า
> `research`
> `proposal`
> ไอเดีย

ปัญหาคือไฟล์ในโฟลเดอร์ธรรมดามักไม่คุยกันเอง

แต่ใน Obsidian โน้ตแต่ละไฟล์สามารถเชื่อมถึงกันได้ เช่น:

- โน้ตประชุมวันนี้ link ไปหาโปรเจกต์ที่เกี่ยวข้อง
- โน้ตลูกค้า link ไปหา pain point ของลูกค้า
- โน้ตบทความที่อ่าน link ไปหา concept ที่เราใช้บ่อย

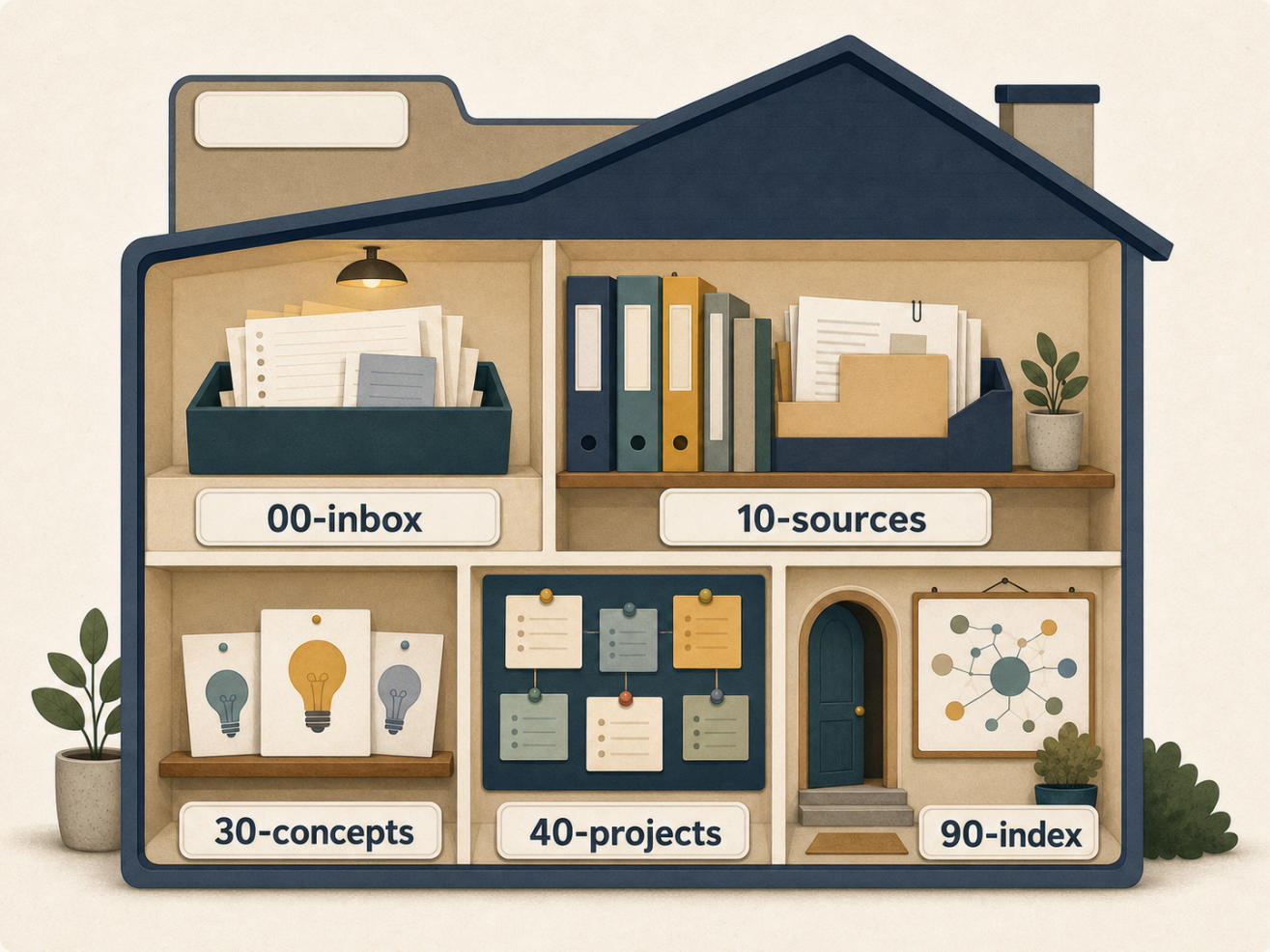
- หน้า project link ไปหา decision, source, todo และ meeting note ทั้งหมด

พอเชื่อมแบบนี้ ความรู้จะไม่ใช่กองไฟล์แยกกัน แต่กลายเป็นแผนที่

และเมื่อถึงวันที่คุณให้ Claude อ่านหรือช่วยสรุป Claude ก็จะเข้าใจง่ายขึ้นว่า “เรื่องนี้เกี่ยวกับอะไร เชื่อมกับอะไร และ source อยู่ตรงไหน”

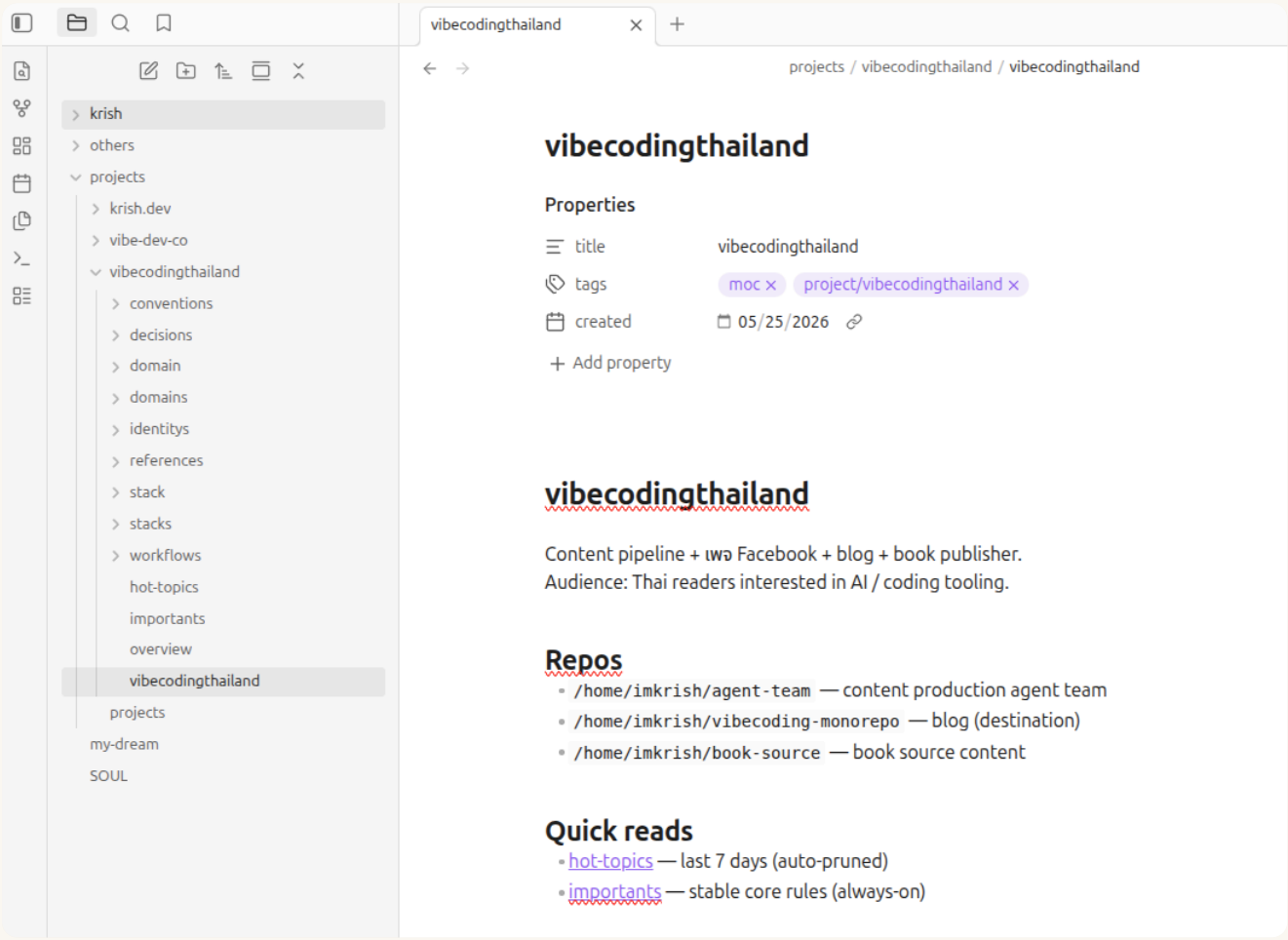
---

2. Vault คือบ้านหนึ่งหลัง



Vault คือบ้านหนึ่งหลังของความรู้ แต่ละห้องคือที่เก็บข้อมูลคนละชนิด

ภาพ: Vault คือบ้านหนึ่งหลังของความรู้ แต่ละห้องคือที่เก็บข้อมูลคนละชนิด



ตัวอย่าง Obsidian ที่ใช้จัด project และ note ให้อยู่เป็นที่

ภาพ: ตัวอย่าง Obsidian ที่ใช้จัด project และ note ให้อยู่เป็นที่

คำแรกที่ต้องรู้คือ **Vault**

Vault คือโฟลเดอร์หลักของความรู้ใน Obsidian

ถ้าคุณทำงานหลายด้าน อาจมีหลาย vault ก็ได้ เช่น:

- vault ส่วนตัว
- vault งานบริษัท
- vault โปรเจกต์หนังสือ
- vault สำหรับเรียนเรื่อง AI

แต่ถ้าเพิ่งเริ่ม อย่าเพิ่งสร้างหลาย vault

ให้เริ่มจาก vault เดียวก่อน เช่น:

> `my-work-wiki`

เหตุผลคือ คนส่วนใหญ่ไม่ได้ล้มเหลวเพราะเครื่องมือไม่พอ แต่ล้มเหลวเพราะเริ่มซับซ้อนเกินไป

เริ่มจากบ้านหลังเดียวให้ใช้งานได้ก่อน แล้วค่อยแยกทีหลัง

ตัวอย่างของหมิว

หมิวเป็น marketing coordinator ในบริษัทเล็ก ๆ

ข้อมูลของหมิวกระจายอยู่หลายที่:

- meeting note อยู่ใน Google Docs
- งานด่วนอยู่ใน LINE
- research คู่แข่งอยู่ใน browser bookmark
- สรุปแคมเปญอยู่ในไฟล์ PDF
- ไอเดีย content อยู่ใน chat กับ Claude

ถ้าหมิวสร้าง vault ชื่อ `mew-work-wiki` แล้วค่อย ๆ ย้ายสิ่งสำคัญเข้ามา หมิวจะเริ่มมีบ้านกลางสำหรับความรู้ของตัวเอง

ไม่ต้องย้ายทุกอย่างในวันเดียว

แค่เริ่มจากสิ่งที่ใช้บ่อยที่สุดก็พอ

---

3. Note คือไฟล์ความรู้หนึ่งชิ้น

ใน Obsidian สิ่งที่คุณเขียนแต่ละหน้าเรียกว่า **note**

Note ที่ดีไม่จำเป็นต้องยาว

บาง note อาจมีแค่ 5 บรรทัด แต่ถ้าหัวข้อชัดและหาเจอ ก็มีค่ากว่าเอกสาร 20 หน้าที่ไม่มีใครเปิดอ่าน

หลักง่าย ๆ คือ:

> หนึ่ง note ควรมีหนึ่งเรื่องหลัก

ตัวอย่าง note ที่ดี:

> คำถามที่ลูกค้าถามบ่อยเรื่องคอร์ส AI
> สรุป meeting กับทีม sales 2026-05-25
> Pain point ของเจ้าของธุรกิจ SME
> ไอเดีย webinar เดือนมิถุนายน
> คู่แข่ง: บริษัท ABC

ตัวอย่าง note ที่ไม่ดี:

> รวมทุกอย่าง
> โน้ตทั่วไป
> AI
> งาน
> ไอเดียหลายเรื่อง

ชื่อ note ที่ชัดมีผลมาก เพราะทั้งคุณและ Claude จะอ่าน context ได้เร็วขึ้น

ถ้าชื่อไฟล์คือ `โน้ตใหม่ 12` คุณเองยังไม่รู้ว่าข้างในคืออะไร Claude ก็ต้องเดาเหมือนกัน

---

4. Markdown คือข้อความธรรมดาที่จัดระเบียบได้

Obsidian ใช้ไฟล์ Markdown เป็นหลัก

ไม่ต้องตกใจ คำว่า Markdown ไม่ใช่เรื่อง code

ให้คิดว่า Markdown คือการพิมพ์ข้อความธรรมดา แต่มีสัญลักษณ์ง่าย ๆ เพื่อบอกว่าอะไรคือหัวข้อ ลิสต์ หรือ checkbox

ตัวอย่าง:

```markdown
# หัวข้อใหญ่

## หัวข้อย่อย

- ประเด็นที่ 1
- ประเด็นที่ 2
- ประเด็นที่ 3

- [ ] งานที่ต้องทำ
- [x] งานที่ทำแล้ว
```

แค่นี้พอสำหรับ 80% ของการใช้ Obsidian ในเล่มนี้

ข้อดีของ Markdown คือ:

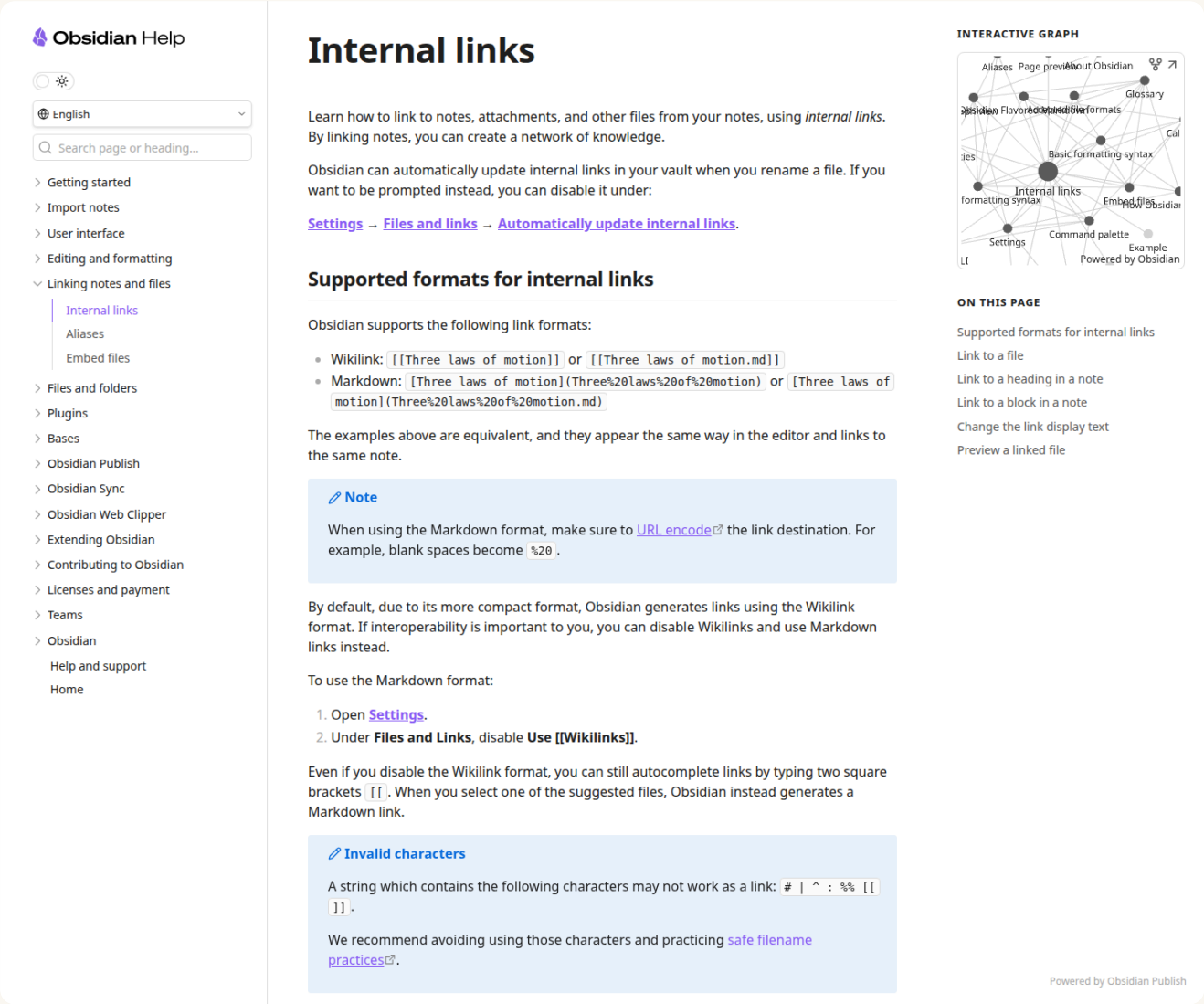
- อ่านได้แม้ไม่เปิด Obsidian
- copy ไปให้ Claude อ่านง่าย
- เก็บเป็นไฟล์ธรรมดา ไม่ติดอยู่ใน format แปลก ๆ
- ย้ายไปใช้เครื่องมืออื่นได้ง่ายกว่าเอกสารที่ล็อกอยู่ในระบบเดียว

สำหรับคนทำงานทั่วไป นี่คือเหตุผลใหญ่ที่สุดที่ Obsidian เหมาะกับ LLM Wiki

เพราะ AI ไม่ต้องพยายามถอดรหัสเอกสารซับซ้อน มันอ่าน text ที่มีโครงสร้างชัดได้เลย

---

5. Link คือเส้นทางระหว่างความรู้



หน้า Obsidian Help เรื่อง Internal links: หัวใจคือการทำให้ note คุยกันได้

ภาพ: หน้า Obsidian Help เรื่อง Internal links: หัวใจคือการทำให้ note คุยกันได้

หัวใจของ Obsidian ไม่ใช่การจดเยอะ

หัวใจคือการเชื่อมโน้ต

ใน Obsidian คุณสามารถ link ไปหา note อื่นได้ด้วยรูปแบบประมาณนี้:

```
MARKDOWN

[[Pain point ของเจ้าของธุรกิจ SME]]
[[ไอเดีย webinar เดือนมิถุนายน]]
[[คู่แข่ง: บริษัท ABC]]
```

ไม่ต้องจำ syntax มาก แค่รู้ว่าการใส่วงเล็บเหลี่ยมคู่คือการบอกว่า “โน้ตนี้เกี่ยวข้องกับอีกโน้ตหนึ่ง”

ตัวอย่าง:

ใน note ชื่อ `สรุป meeting กับทีม sales 2026-05-25` อาจเขียนว่า:

```
MARKDOWN

ลูกค้าถามเรื่อง [[ราคาแพ็กเกจอบรม AI]] บ่อยขึ้น
ทีม sales บอกว่า pain point หลักคือ [[ผู้บริหารไม่รู้จะเริ่มใช้ AI ตรงไหน]]
ควรเอาไปทำ content ใน [[ไอเดีย webinar เดือนมิถุนายน]]
```

พอเขียนแบบนี้ คุณไม่ได้แค่บันทึกประชุม

คุณกำลังสร้างทางเดินให้ความรู้เดินถึงกัน

วันต่อมา ถ้าคุณเปิด note เรื่อง `ผู้บริหารไม่รู้จะเริ่มใช้ AI ตรงไหน` คุณจะเห็นว่ามันเกี่ยวกับ meeting ไหน content ไหน และลูกค้ากลุ่มไหน

นี่คือจุดที่ Obsidian เริ่มต่างจากแอปจดโน้ตทั่วไป

---

6. Backlink คือ “ใครพูดถึงเรื่องนี้บ้าง”

ถ้า link คือการชี้จาก note A ไป note B

Backlink คือการดูย้อนกลับว่า note ไหนบ้างที่ชี้มาหา note นี้

ตัวอย่าง:

คุณมี note ชื่อ:

> `ผู้บริหารไม่รู้จะเริ่มใช้ AI ตรงไหน`

เมื่อเปิด note นี้ Obsidian สามารถบอกได้ว่า note นี้ถูกพูดถึงจาก:

- meeting กับทีม sales
- research จากบทความหนึ่ง
- ไอเดีย webinar

- FAQ หน้าขายคอร์ส
- script สำหรับ video สั้น

สำหรับคนทำงาน นี่มีประโยชน์มาก เพราะมันตอบคำถามว่า:

> “เรื่องนี้โผล่มาจากที่ไหนบ้าง?”

และสำหรับ Claude มันช่วยให้ context ไม่ขาด

แทนที่จะให้ Claude อ่าน note เดี่ยว ๆ คุณสามารถให้มันอ่าน note หลัก พร้อม note ที่เชื่อมโยงกัน แล้วถามว่า:

> `จาก note เหล่านี้ ช่วยสรุป pattern ของลูกค้าที่ถามเรื่องการเริ่มใช้ AI ในบริษัทให้หน่อย`

ผลลัพธ์จะดีกว่าการถามแบบลอย ๆ มาก

---

7. Tag ใช้เป็นป้ายกำกับ อย่าใช้เป็นระบบทั้งชีวิต

Tag คือป้ายกำกับ เช่น:

> #ลูกค้า
> #ไอเดีย
> #ประชุม
> #ต้องทำ
> #รอข้อมูล

Tag มีประโยชน์ แต่ก็เป็นกับดัก

คนเริ่มใช้ Obsidian ใหม่ ๆ มักสร้าง tag เยอะเกินไป เช่น:

```
#marketing
#content
#contentidea
#idea-content
#ไอเดียcontent
#content-marketing
```

สุดท้าย tag เยอะจนหาอะไรไม่เจอ

กติกาง่าย ๆ คือ:

> ใช้ tag สำหรับสถานะหรือหมวดกว้าง ๆ ไม่ใช่แทนชื่อเรื่อง

ตัวอย่าง tag ที่พอใช้ได้:

```
#inbox
#source
#idea
#todo
#waiting
#review
```

ส่วนเรื่องเฉพาะ เช่น “AI สำหรับเจ้าของธุรกิจ” หรือ “ปัญหาทีม sales” ให้ทำเป็น note แล้ว link ไปหา ดีกว่าใช้ tag ยาว ๆ

สรุปสั้น ๆ:

- tag = ป้ายกำกับกว้าง
- link = ความสัมพันธ์ระหว่างเรื่อง
- note = ความรู้หนึ่งชิ้น

อย่าให้ tag ทำงานแทนทุกอย่าง

---

8. Property คือช่องข้อมูลสั้น ๆ บนหัวโน้ต

Property คือข้อมูลสั้น ๆ ที่ติดอยู่กับ note

ให้คิดเหมือนช่องในแบบฟอร์ม เช่น:

```
type: source
status: draft
date: 2026-05-25
source: https://example.com/article
project: ai-training-campaign
```

คนทั่วไปไม่จำเป็นต้องใช้ property เยอะ

สำหรับเล่มนี้ แนะนำเริ่มจาก 4 ช่องพอ:

```yaml
type: source
status: inbox
date: 2026-05-25
source:
```

ใช้เมื่อไหร่?

- ถ้า note นี้มาจากบทความ ให้ใส่ `type: source`
- ถ้ายังไม่ได้จัด ให้ใส่ `status: inbox`
- ถ้าเป็น note ของโปรเจกต์ ให้ใส่ `type: project`
- ถ้าเป็นแนวคิด ให้ใส่ `type: concept`

ข้อดีคือ เมื่อ note เยอะขึ้น คุณจะกรองและจัดกลุ่มได้ง่ายขึ้น

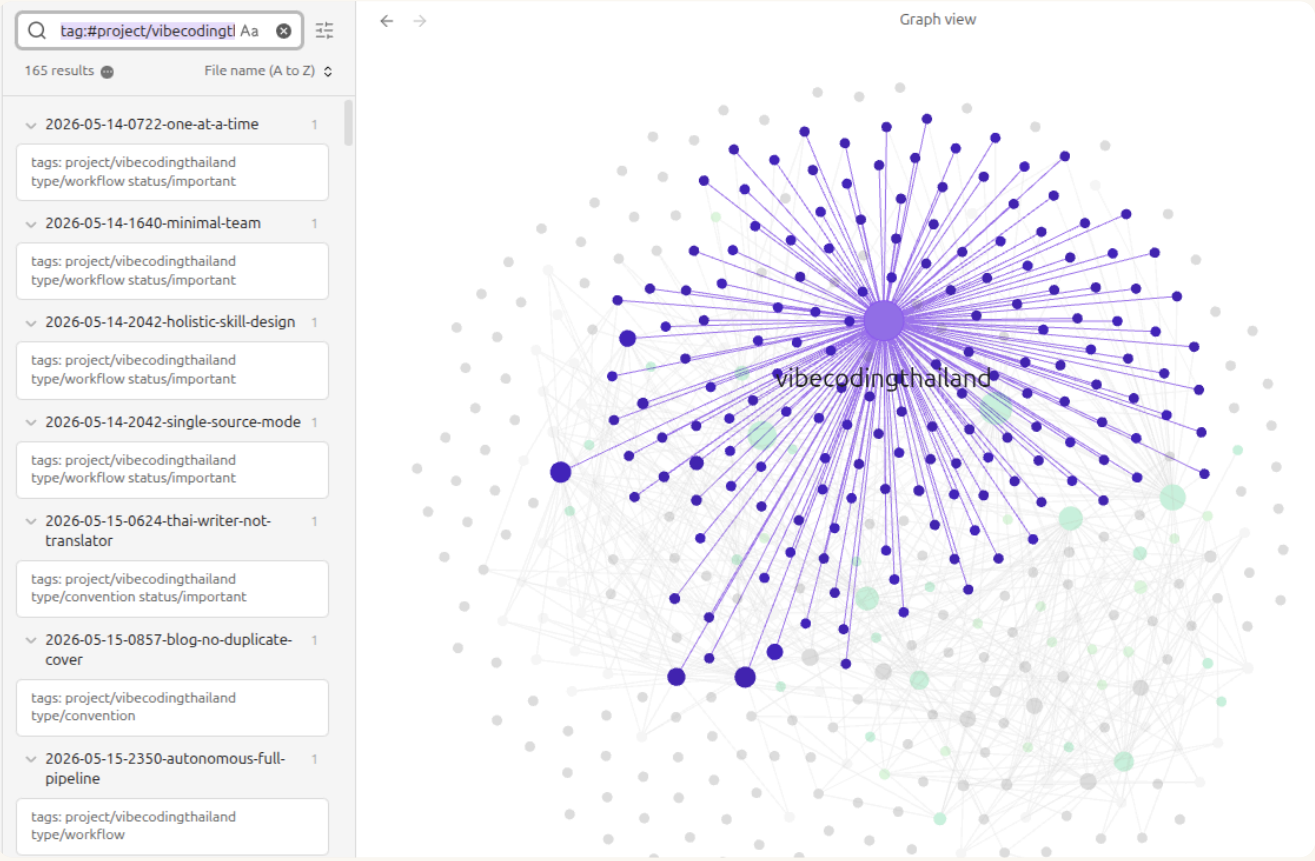
แต่ข้อควรระวังคือ อย่าเริ่มจาก property 20 ช่อง เพราะจะทำให้คุณเลิกใช้ระบบตั้งแต่สัปดาห์แรก

เริ่มน้อยก่อน

---

9. Graph view สวย แต่ไม่ใช่เป้าหมาย

Obsidian มี Graph view ที่แสดงโน้ตเป็นจุด ๆ แล้วเชื่อมกันเป็นเส้น



ตัวอย่าง Graph view ใน Obsidian ที่ทำให้เห็น note ที่เกี่ยวกันเป็นภาพรวม

ภาพ: ตัวอย่าง Graph view ใน Obsidian ที่ทำให้เห็น note ที่เกี่ยวกันเป็นภาพรวม

มันดูเท่ และช่วยให้เห็นภาพรวมว่า note ไหนเชื่อมกับ note ไหน

แต่สำหรับเล่มนี้ ขอให้จำไว้ว่า:

> Graph view เป็นผลพลอยได้ ไม่ใช่เป้าหมาย

เป้าหมายจริงคือ:

- หา note เจอ
- อ่านแล้วเข้าใจ
- รู้ว่า source อยู่ไหน

- เอา note ไปให้ Claude ช่วยต่อได้
- สรุปงานและตัดสินใจได้เร็วขึ้น

ถ้าคุณมัวแต่ทำ graph ให้สวย แต่ note ข้างในอ่านไม่รู้เรื่อง ระบบนี้ก็ไม่ช่วยอะไร

ให้ใช้ Graph view แค่เช็กภาพรวม เช่น:

- มี note ไหนโดดเดี่ยวเกินไปไหม
- มีเรื่องไหนถูกพูดถึงบ่อยไหม
- โปรเจกต์ไหนเชื่อมกับ source ไหนบ้าง

แต่ไม่ต้องทำให้มันสวย

---

10. Canvas, Bases, Web Clipper ใช้เมื่อไหร่

Obsidian มี feature อีกหลายอย่าง แต่คนเริ่มต้นควรรู้แค่ภาพรวม

Canvas

Canvas คือพื้นที่วาง note, card, รูป และ link แบบกระดานภาพ

เหมาะกับ:

- วางโครงบทความ
- map ไอเดียแคมเปญ
- จัด flow การทำงาน
- วางภาพรวมโปรเจกต์

ไม่เหมาะกับการเก็บทุกอย่าง เพราะถ้าทุกเรื่องอยู่บน canvas จะรกเร็ว

ใช้ canvas เมื่อคุณต้อง “เห็นภาพรวม”

Bases

Bases คือวิธีดู note เป็นตารางหรือมุมมองแบบจัดกลุ่ม เหมาะกับ note ที่มี property ชัด

เหมาะกับ:

- list บทความที่เก็บไว้
- list ลูกค้า
- list ไอเดีย content
- list โปรเจกต์
- list source ที่รออ่าน

ไม่ต้องใช้ตั้งแต่วันแรกก็ได้

ถ้าคุณยังมี note ไม่ถึง 50 ไฟล์ ใช้ folder + search + link ก่อนก็พอ

Web Clipper

Web Clipper ใช้เก็บข้อมูลจากเว็บเข้ามาใน Obsidian

เหมาะกับ:

- บทความที่อยากเก็บไว้เป็น source
- หน้าเว็บคู่แข่ง
- reference สำหรับงาน
- ไอเดียที่เจอระหว่าง research

แต่ต้องระวังอย่างหนึ่ง:

> clip เยอะไม่ได้แปลว่ารู้เยอะ

ถ้าเก็บเว็บ 100 หน้า แต่ไม่เคยสรุป ไม่เคย link ไม่เคยใช้ มันก็เป็นแค่กองของที่ยังไม่ได้จัด

ให้ใช้ Web Clipper เป็นทางเข้า ไม่ใช่ปลายทาง

---

11. โครงสร้าง Vault แบบเริ่มเร็ว

นี่คือโครงสร้างที่แนะนำสำหรับคนทั่วไป:

```
/my-wiki
  /00-inbox
  /10-sources
  /20-notes
  /30-concepts
  /40-projects
  /90-index
```

`/00-inbox`

ที่พักของทุกอย่างที่ยังไม่ได้จัด

ใส่ได้ทั้ง link, note ด่วน, meeting note, idea, ข้อความที่ copy มา

กติกาคือ inbox ต้องถูกล้างเป็นระยะ ไม่ใช่กลายเป็นถังขยะถาวร

`/10-sources`

เก็บ note ที่มาจากแหล่งอ้างอิง เช่น:

- บทความ
- PDF
- meeting transcript
- หน้าเว็บคู่แข่ง
- official docs
- interview note

Source note ควรบอกชัดว่าเอามาจากไหน

`/20-notes`

เก็บ note ทั่วไปที่ยังไม่ใช่ concept ชัด ๆ เช่น:

- สรุปประชุม
- ไอเดียเบื้องต้น
- observation
- checklist

`/30-concepts`

เก็บแนวคิดที่ใช้ซ้ำ เช่น:

- Pain point ของลูกค้า SME
- AI readiness
- Source-first thinking
- Weekly review

Concept note คือหัวใจของ LLM Wiki เพราะมันทำให้ความรู้ไม่กระจายอยู่แค่ใน source

`/40-projects`

เก็บหน้าโปรเจกต์ เช่น:

- แคมเปญ webinar เดือนมิถุนายน
- หนังสือ Claude + Obsidian
- ปรับ onboarding ลูกค้าใหม่
- ทำ knowledge base ทีม sales

Project note ควรเป็นหน้ารวมทุกอย่างของโปรเจกต์นั้น

`/90-index`

เก็บหน้า index หรือหน้า home เช่น:

- Home
- Index ลูกค้า
- Index โปรเจกต์
- Index แหล่งอ้างอิง
- Index ไอเดีย content

Index คือประตูหน้าบ้าน

ถ้าไม่มี index ทั้งคุณและ Claude จะต้องเดินหาเองทุกครั้ง

---

12. หน้า Home แรกของคุณ

สร้าง note ชื่อ `Home.md` หรือ `เริ่มที่นี่.md`

ใส่เนื้อหาง่าย ๆ แบบนี้:

```markdown
# Home

นี่คือ LLM Wiki ส่วนตัวของฉัน

## ใช้ทำอะไร

- เก็บ source สำคัญ
- สรุป meeting และไอเดีย
- เชื่อมโยงความรู้สำหรับให้ Claude ช่วยต่อ
- ทำ project brief และ decision log

## โฟลเดอร์หลัก

- [[00-inbox]]: ของที่ยังไม่ได้จัด
- [[10-sources]]: แหล่งข้อมูลและ reference
- [[20-notes]]: โน้ตทั่วไป
- [[30-concepts]]: แนวคิดที่ใช้ซ้ำ
- [[40-projects]]: งานและโปรเจกต์
- [[90-index]]: หน้า index สำคัญ

## กติกาของ Wiki นี้

1. หนึ่ง note หนึ่งเรื่องหลัก
2. ถ้ามี source ให้ใส่ source
3. เรื่องสำคัญต้อง link กลับมาหา project หรือ index
4. inbox ต้องถูกล้างทุกสัปดาห์
5. Claude ช่วยจัดได้ แต่ฉันเป็นคนตรวจสุดท้าย
```

หน้า Home ไม่ต้องสมบูรณ์

หน้าที่ของมันคือทำให้คุณเริ่มจากจุดเดียวได้เสมอ

---

13. เริ่มวันนี้ใน 20 นาที

ถ้าคุณอยากเริ่มทันที ให้ทำตามนี้:

1. สร้าง vault ใหม่ชื่อ `my-wiki`
2. สร้าง folder 6 อัน:

```
00-inbox
10-sources
20-notes
30-concepts
40-projects
90-index
```

3. สร้าง note ชื่อ `Home`
4. copy template ด้านบนใส่ลงไป
5. เลือกข้อมูลจริง 1 ชิ้นที่คุณใช้ทำงาน เช่น meeting note หรือบทความหนึ่งหน้า
6. ใส่ไว้ใน `00-inbox`
7. เขียนเพิ่ม 3 บรรทัด:

```markdown
## สรุปสั้น

## เกี่ยวข้องกับ

## ต้องทำต่อ
```

แค่นี้พอ

อย่าเพิ่งติดตั้ง plugin เพิ่ม อย่าเพิ่งจัดสี อย่าเพิ่งทำ graph ให้สวย

เป้าหมายของวันแรกคือ “เริ่มมีบ้าน” ไม่ใช่ “สร้างคฤหาสน์”

สรุปท้ายบท

Obsidian เหมาะกับ LLM Wiki เพราะมันทำสามอย่างได้ดี:

1. เก็บความรู้เป็นไฟล์ที่อ่านง่าย
2. เชื่อมโน้ตเข้าหากันได้

3. ทำให้คุณและ Claude หา context ได้ง่ายขึ้น

สิ่งที่ต้องรู้ในตอนนี้มีแค่นี้:

- Vault = บ้านหนึ่งหลัง
- Note = ความรู้หนึ่งชิ้น
- Markdown = ข้อความธรรมดาที่จัดรูปแบบง่าย
- Link = เส้นทางจาก note หนึ่งไปอีก note หนึ่ง
- Backlink = ใครพูดถึงเรื่องนี้บ้าง
- Tag = ป้ายกำกับกว้าง ๆ
- Property = ช่องข้อมูลสั้น ๆ ของ note
- Graph = ภาพรวม ไม่ใช่เป้าหมาย
- Canvas/Bases/Web Clipper = เครื่องมือเสริม ใช้เมื่อจำเป็น

บทต่อไป เราจะเอา Claude เข้ามาในระบบนี้

ถ้า Obsidian คือบ้านของความรู้ Claude คือผู้ช่วยที่ช่วยอ่านบ้านหลังนี้ สรุปของในบ้าน จัดห้อง และถามกลับว่า “มีอะไรที่ยังขาดอยู่ไหม”

แต่ Claude ไม่ใช่เจ้าของความจริง

เจ้าของความรู้ยังเป็นคุณ

Artifact ท้ายบท: โครง Vault เริ่มต้น

```
/my-wiki
  /00-inbox
  /10-sources
  /20-notes
  /30-concepts
  /40-projects
  /90-index
```

Artifact ท้ายบท: กติกา 5 ข้อ

1. จับก่อน ค่อยจัด
2. หนึ่ง note หนึ่งเรื่องหลัก
3. ถ้ามี source ให้เก็บ source
4. อย่าใช้ tag เยอะเกินไป
5. เรื่องสำคัญควร link กลับไป project หรือ index

อ้างอิงหลักของบทนี้

- Obsidian Help: How Obsidian stores data
- Obsidian Help: Create a vault
- Obsidian Help: Create your first note
- Obsidian Help: Internal links
- Obsidian Help: Backlinks
- Obsidian Help: Properties
- Obsidian Help: Graph view
- Obsidian Help: Canvas
- Obsidian Help: Bases
- Obsidian Help: Web Clipper

บทที่ 2

Claude คือผู้ช่วยอ่าน คิด และจัดระบบ ไม่ใช่เจ้าของความจริง

บทที่แล้ว เราสร้าง “บ้านของความรู้” ด้วย Obsidian

บ้านหลังนี้มี vault, note, link, backlink, tag, property และหน้า Home

คำถามต่อมาคือ:

> แล้ว Claude เข้ามาอยู่ตรงไหนในบ้านหลังนี้?

คำตอบสั้น ๆ คือ:

Claude ไม่ใช่บ้าน

Claude คือผู้ช่วยที่เข้ามาอ่านของในบ้าน จัดของให้เป็นหมวด สรุปสิ่งที่กระจัดกระจาย ถามกลับเมื่อข้อมูลยังขาด และช่วยร่างงานต่อจากความรู้ที่เรามี

แต่ Claude ไม่ใช่เจ้าของความจริง

นี่คือ mindset สำคัญของทั้งเล่ม

ถ้าคุณให้ Claude เดา มันอาจตอบมั่นใจทั้งที่ผิด

แต่ถ้าคุณให้ Claude อ่าน note ที่มี source ชัด มี context ชัด และมีคำสั่งชัด มันจะกลายเป็นผู้ช่วยที่มีประโยชน์มาก

บทนี้จะอธิบายว่า Claude ช่วยงาน LLM Wiki ได้ยังไง โดยไม่ต้องพูดเรื่อง code

---

1. Claude ไม่ได้ “จำทุกอย่าง” แทนเรา

หลายคนเริ่มใช้ AI แล้วหวังว่า AI จะจำทุกเรื่องให้หมด

คุยวันนี้ พรุ่งนี้ถามต่อ เดือนหน้ากลับมาถามอีกที แล้วหวังว่ามันจะรู้ทันทีว่าเราหมายถึงอะไร

ในชีวิตจริง มันไม่ง่ายแบบนั้น

Claude มีความสามารถหลายแบบ เช่น:

- อ่านไฟล์ที่เราอัปโหลด
- ใช้ context ใน project
- ช่วยสร้าง artifact หรือเอกสารที่ใช้ต่อได้
- ค้นเว็บหรือทำ research ในบางกรณี
- ใช้ chat history หรือ memory ตาม feature และ setting ที่เปิดใช้งาน

แต่ไม่ว่าฟีเจอร์จะดีแค่ไหน คุณไม่ควรฝากความรู้ทั้งหมดไว้ใน chat อย่างเดียว

เพราะ chat มีปัญหา 3 อย่าง:

1. ยาวขึ้นเรื่อย ๆ จนหาแก่นไม่เจอ
2. มีทั้ง fact, idea, draft, ความเข้าใจผิด และการแก้ไขปนกัน
3. ยากที่จะเอากลับมาใช้เป็นระบบในงานอื่น

Obsidian จึงทำหน้าที่เป็น “ความจำที่เราคุมเอง”

Claude ทำหน้าที่เป็น “ผู้ช่วยที่มาอ่านความจำนี้แล้วช่วยคิดต่อ”

นี่ต่างกันมาก

---

2. งานที่ Claude ช่วยได้จริงใน Wiki

สำหรับ LLM Wiki ของคนทำงานทั่วไป Claude ช่วยได้ 5 งานหลัก

1. อ่าน source แล้วสรุป

คุณเอาบทความ meeting note PDF หรือข้อความจากเว็บให้ Claude อ่าน แล้วให้มันสรุปเป็น note ที่สั้นและใช้ต่อได้

ตัวอย่าง:

```
ช่วยอ่าน source นี้แล้วสรุปเป็นโน้ต Obsidian
ขอ 5 ส่วน:
1. สรุปสั้น 5 บรรทัด
2. key facts
3. ประเด็นที่เกี่ยวกับงานของฉัน
4. คำถามที่ควรค้นต่อ
5. tag หรือ link ที่ควรเชื่อม
```

นี่ช่วยลดเวลาจาก “อ่านแล้วจดเองทั้งหมด” เป็น “ให้ Claude ช่วยร่าง แล้วเราอ่านตรวจ”

จุดสำคัญคืออย่าใช้คำว่า “สรุปให้หน่อย” แบบลอย ๆ

ให้บอก format ที่ต้องการเสมอ

2. แตก note

บางครั้ง source หนึ่งชิ้นมีหลายเรื่องปนกัน

เช่น meeting note หนึ่งหน้าอาจมีทั้ง:

- ลูกค้าบ่นเรื่องราคา
- ทีม sales ขอ FAQ
- ไอเดีย webinar
- task ที่ต้องทำ
- decision เรื่อง budget

ถ้าเก็บทั้งหมดไว้ใน note เดียว มันจะรก

Claude ช่วยแตกออกเป็น note ย่อยได้

```
ช่วยดู note นี้แล้วบอกว่าควรแตกเป็น note ย่อยอะไรบ้าง
แยกเป็น:
- source note
- concept note
- project note
- decision note
- todo
อย่าแต่งข้อมูลใหม่ ใช้เฉพาะสิ่งที่มีใน note นี้
```

3. เชื่อมโยง note

เมื่อมี note หลายไฟล์ สิ่งที่คนมักลืมคือการ link

Claude ช่วยเสนอได้ว่า note นี้ควรเชื่อมกับเรื่องไหน

ตัวอย่าง:

```
จาก note นี้ ช่วยเสนอ internal links สำหรับ Obsidian
กติกา:
- link เฉพาะเรื่องที่ควรเป็น note จริง
- อย่าสร้าง link ให้คำทั่วไปเกินไป
- ถ้าไม่แน่ใจ ให้ใส่ไว้ในหัวข้อ "อาจสร้าง note เพิ่ม"
```

ผลลัพธ์ที่ดีไม่ใช่ link เยอะที่สุด

แต่คือ link ที่ทำให้อนาคตคุณและ Claude เดินทางใน wiki ได้ง่ายขึ้น

4. ถามกลับเพื่อเติมช่องว่าง

Claude ไม่ควรมีหน้าที่ตอบอย่างเดียว

บางครั้งคำตอบที่ดีที่สุดคือคำถาม

เช่นคุณให้ Claude อ่าน note เกี่ยวกับแผน campaign แล้วมันอาจถามกลับว่า:

- กลุ่มเป้าหมายหลักคือใคร
- success metric คืออะไร
- deadline คือเมื่อไหร่
- มี source จากลูกค้าจริงไหม
- decision ล่าสุดคืออะไร

นี่สำคัญมาก เพราะ LLM Wiki ที่ดีไม่ได้มีแค่ข้อมูลเยอะ แต่ต้องเห็นด้วยว่า “ข้อมูลอะไรยังขาด”

Prompt ที่ใช้ได้:

> ช่วยอ่าน `note` นี้แล้วทำ 2 อย่าง:
> 1. สรุปสิ่งที่เรารู้แล้ว
> 2. ถามคำถามที่ควรตอบเพิ่มก่อนเอา `note` นี้ไปใช้ทำงานจริง
>
> ห้ามเดาคำตอบแทนฉัน
> ถ้าข้อมูลขาด ให้บอกว่าขาด

5. ช่วยทำ draft จากความรู้ที่มี

เมื่อ wiki เริ่มมี source และ concept มากขึ้น Claude จะช่วยร่างงานได้ดีขึ้น เช่น:

- brief แคมเปญ
- outline บทความ
- FAQ ลูกค้า
- proposal
- script presentation
- meeting summary
- decision log

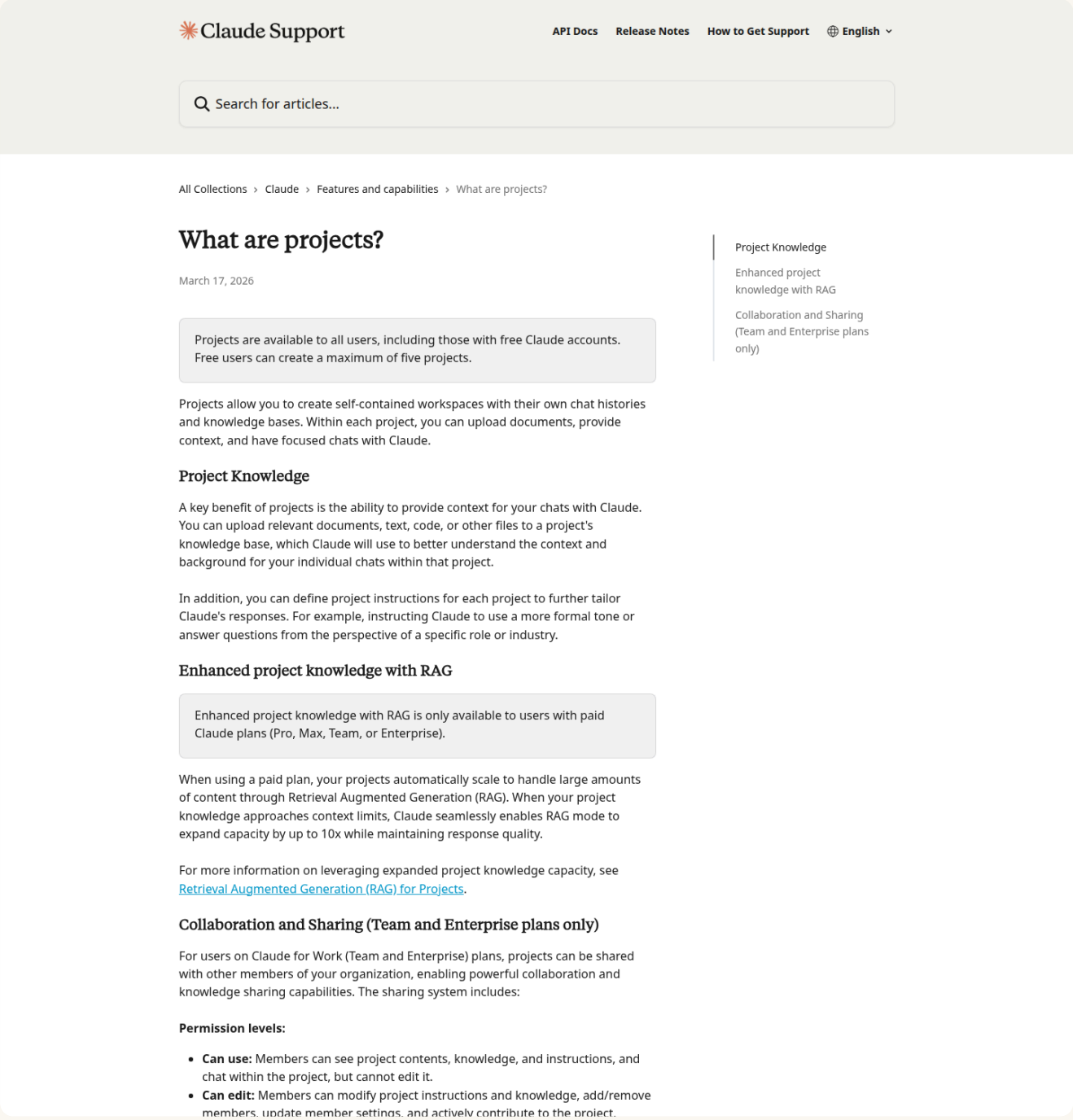
แต่ต้องย้ำอีกครั้ง:

Claude ช่วยร่าง ไม่ใช่ตัดสินใจแทน

งานสุดท้ายยังต้องผ่านสายตาคุณ

---

3. Claude Projects ใช้ยังไงกับ Obsidian



หน้า Claude Support เรื่อง Projects: ใช้แยก context ตามงาน แต่ไม่แทน Obsidian ทั้ง vault

ภาพ: หน้า Claude Support เรื่อง Projects: ใช้แยก context ตามงาน แต่ไม่แทน Obsidian ทั้ง vault

ถ้าคุณใช้ Claude ผ่านเว็บหรือแอป หนึ่งในฟีเจอร์สำคัญคือ **Projects**

ให้คิดว่า Project คือพื้นที่ทำงานเฉพาะเรื่องใน Claude

เช่น:

- Project สำหรับแคมเปญ marketing
- Project สำหรับหนังสือเล่มนี้
- Project สำหรับ knowledge base ทีม sales
- Project สำหรับงาน research คู่แข่ง

ใน Project คุณสามารถใส่ context, instruction และไฟล์ที่เกี่ยวข้อง เพื่อให้ Claude ตอบโดยเข้าใจงานนั้นมากขึ้น

สำหรับคนใช้ Obsidian วิธีคิดง่าย ๆ คือ:

> Obsidian คือคลังหลัก ส่วน Claude Project คือห้องทำงานชั่วคราวของเรื่องที่กำลังทำ

ไม่จำเป็นต้องเอา vault ทั้งก้อนยัดเข้า Claude

ให้เลือกเฉพาะ note ที่เกี่ยวข้องกับงานนั้น

ตัวอย่าง:

ต่อจากงาน FAQ หน้าเว็บคอร์ส AI หมิวกำลังเตรียม idea สำหรับ webinar เรื่อง “AI สำหรับเจ้าของธุรกิจ”

ใน Obsidian หมิวมี note เหล่านี้:

```
/10-sources/บทความ official เรื่อง Claude Projects.md
/20-notes/คำถามจากทีม sales เรื่อง AI.md
/30-concepts/Pain point ของเจ้าของธุรกิจ SME.md
/40-projects/Webinar AI เดือนมิถุนายน.md
/90-index/Index content ideas.md
```

หมิวไม่ต้องส่งทุก note ใน vault ให้ Claude

หมิวเลือก 4–5 note ที่เกี่ยวข้อง แล้วบอก Claude ว่า:

```
นี่คือ context สำหรับ idea webinar ที่ต่อยอดจากงาน FAQ
ช่วยอ่าน note เหล่านี้ก่อน แล้วตอบโดยยึดข้อมูลจาก note เป็นหลัก
ถ้าจะเสนอไอเดียใหม่ ให้แยกเป็น "ข้อเสนอ" ไม่ปนกับ fact
อย่าเขียนเหมือน webinar นี้เป็น decision แล้ว
```

แบบนี้ Claude จะทำงานแม่นกว่า chat เปล่า ๆ มาก

---

4. Files, Project Knowledge, Artifacts ต่างกันยังไงแบบง่าย

ไม่ต้องจำศัพท์เยอะ แต่ควรเข้าใจภาพรวม

Files

Files คือไฟล์ที่คุณส่งให้ Claude อ่านใน chat หรือ project

เช่น PDF, TXT, CSV, DOCX, HTML, JSON หรือรูปภาพบางประเภท

ใช้เมื่อคุณมีเอกสารที่อยากให้ Claude อ่านหรือสรุป

ตัวอย่าง:

- อัปโหลด meeting transcript
- อัปโหลด PDF proposal
- อัปโหลด export จาก Obsidian note
- อัปโหลดภาพ screenshot ที่อยากให้ช่วยอ่าน

Project Knowledge

Project Knowledge คือไฟล์หรือข้อความที่อยู่กับ Claude Project นั้น ๆ เพื่อใช้เป็น context ในหลาย chat ภายใน project เดียวกัน

ใช้เมื่อเรื่องนั้นต้องคุยหลายรอบ

เช่น project งาน webinar มี:

- audience profile
- product info
- key message
- source สำคัญ
- FAQ เดิม

แทนที่จะอัปโหลดใหม่ทุก chat คุณใส่ไว้ใน project knowledge แล้วคุยต่อใน project นั้น

Artifacts

Artifacts คือชิ้นงานที่ Claude สร้างแยกออกมาเป็นหน้าต่างหรือไฟล์ที่แก้ต่อได้ เช่น document, markdown, diagram, table หรือ content draft

สำหรับเล่มนี้ ให้ใช้ artifact เป็น “พื้นที่ร่างงาน”

ตัวอย่าง:

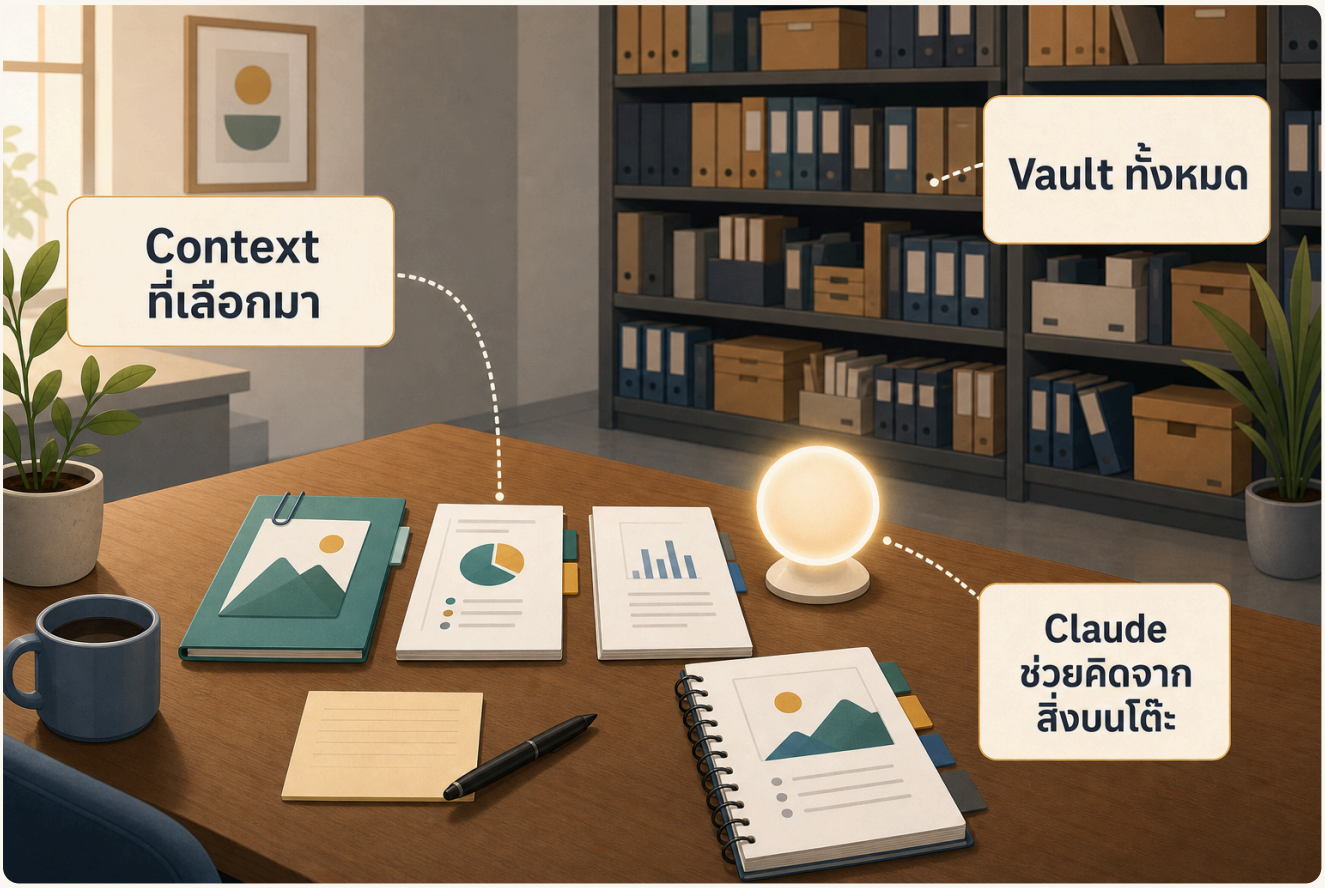
- Claude ร่าง `Project Brief` ให้
- Claude ทำ `FAQ ลูกค้า` ให้
- Claude ทำ `Weekly Review Summary` ให้
- Claude สร้าง template Markdown ให้เอาไปวางใน Obsidian

จำง่าย ๆ:

```
Files = ของที่ให้ Claude อ่าน
Project Knowledge = context ประจำเรื่อง
Artifacts = ชิ้นงานที่ Claude ร่างให้ใช้ต่อ
Obsidian = บ้านหลักของความรู้
```

---

5. Context window คือโต๊ะทำงาน ไม่ใช่โกดัง



Context window เหมือนโต๊ะทำงาน ควรวางเฉพาะเอกสารที่เกี่ยวกับโจทย์ตอนนั้น

ภาพ: Context window เหมือนโต๊ะทำงาน ควรวางเฉพาะเอกสารที่เกี่ยวกับโจทย์ตอนนั้น

มีคำหนึ่งที่คุณอาจเจอบ่อยคือ **context window**

ไม่ต้องเข้าใจเชิงเทคนิค

ให้คิดว่า context window คือ “โต๊ะทำงานของ Claude ในรอบนั้น”

ถ้าคุณเอาเอกสารทั้งหมดในชีวิตวางบนโต๊ะ โต๊ะจะรก

Claude อาจอ่านไม่ครบ หรือจับประเด็นผิด

แต่ถ้าคุณเลือกเฉพาะเอกสารที่เกี่ยวข้อง วางให้เป็นชุด พร้อมบอกว่าแต่ละชิ้นคืออะไร Claude จะช่วยได้ดีขึ้น

กติกาง่าย ๆ:

1. อย่าส่งทุกอย่าง

2. ส่งเฉพาะ note ที่เกี่ยวข้อง
3. บอก Claude ว่า note ไหนคือ source, note ไหนคือ draft, note ไหนคือ decision
4. ถ้าเอกสารยาว ให้แบ่งเป็นชุด
5. ให้ Claude สรุปกลับก่อนเริ่มงานใหญ่

Prompt ที่ใช้ได้:

> ก่อนเริ่มทำงาน ช่วยอ่าน context ที่ฉันให้ แล้วสรุปกลับมาก่อนว่า:
> 1. งานนี้เกี่ยวกับอะไร
> 2. source หลักคืออะไร
> 3. decision ที่มีแล้วคืออะไร
> 4. ข้อมูลอะไรยังขาด
> 5. ถ้าคุณจะทำงานต่อ คุณเข้าใจโจทย์ว่าอะไร
>
> อย่าเพิ่งร่างงานจนกว่าฉันจะยืนยัน

นี่ช่วยกันพลาดได้มาก

เพราะหลายครั้ง Claude ตอบผิดตั้งแต่ต้น ไม่ใช่เพราะมันเขียนไม่เก่ง แต่เพราะมันเข้าใจ context ผิด

---

6. Web Search และ Research ใช้เมื่อไหร่

Claude บางโหมดสามารถค้นเว็บหรือทำ research ได้

ฟีเจอร์นี้มีประโยชน์มาก แต่ต้องใช้ให้ถูก

สำหรับ LLM Wiki ให้แยกง่าย ๆ แบบนี้:

ใช้ Obsidian เมื่อ...

- เป็นความรู้ของคุณเอง
- เป็นข้อมูลบริษัทหรืองานที่คุณเก็บไว้
- เป็น decision ที่เกิดขึ้นในทีม
- เป็น note ที่ต้องใช้ซ้ำ
- เป็น source ที่คุณอยากควบคุมเอง

ใช้ Web Search / Research เมื่อ...

- ต้องการข้อมูลปัจจุบัน
- ต้องการดู official source ล่าสุด
- ต้องการตรวจว่าข้อมูลใน wiki เก่าไปหรือยัง
- ต้องการหา source ใหม่มาเติม
- ต้องการเปรียบเทียบข้อมูลจากหลายแหล่ง

อย่าใช้ web search แทนการจัด wiki

และอย่าใช้ wiki แทนการตรวจข้อมูลล่าสุด

สองอย่างนี้ทำงานคู่กัน

ตัวอย่าง:

หมิวมี note เรื่อง “Claude Projects” ใน Obsidian ที่จดไว้เมื่อหลายเดือนก่อน

ก่อนนำไปใช้สอนทีม หมิวอาจถาม Claude:

> ช่วยตรวจข้อมูลล่าสุดจาก official source เรื่อง Claude Projects
> แล้วเทียบกับ note เดิมของฉัน
> แยกผลลัพธ์เป็น:
> - ยังถูกต้อง
> - ควรอัปเดต
> - ไม่แน่ใจ ต้องตรวจเอง
> - source official ที่ใช้ตรวจ

แบบนี้ wiki จะไม่กลายเป็นกองข้อมูลเก่า

---

7. อย่าให้ Claude เขียนทับความจริง

นี่คือข้อควรระวังที่สำคัญที่สุดของบทนี้

Claude เขียนดีมากจนบางครั้งเราลืมถามว่า “จริงไหม?”

สำหรับ LLM Wiki ให้แยกเนื้อหา 3 แบบเสมอ:

> Fact = สิ่งที่ source บอกจริง
> Inference = ข้อสรุปที่เราคิดจาก fact
> Idea = ไอเดียที่ยังไม่ได้พิสูจน์

ตัวอย่าง:

```
MARKDOWN

## Fact
ลูกค้า 8 จาก 10 คนถามว่าคอร์สนี้ต้องรู้โค้ดไหม จาก meeting note วันที่ 2026-05-20

## Inference
คนกลุ่มนี้น่าจะกังวลว่า AI = programming

## Idea
ควรทำ content ชื่อ "ใช้ AI ในธุรกิจโดยไม่ต้องเขียนโค้ด"
```

ถ้าไม่แยกแบบนี้ Claude อาจเอา idea ไปพูดเหมือน fact

และคุณอาจเอา draft ไปใช้จริงโดยไม่รู้ว่ามันยังเป็นแค่ข้อเสนอ

กติกาที่ควรใส่ในทุก prompt คือ:

> แยก fact / inference / idea ให้ชัด
> ถ้าไม่มี source ให้บอกว่าไม่มี source
> อย่าแต่งข้อมูลเพิ่มเพื่อให้คำตอบดูสมบูรณ์

สั้น แต่ช่วยได้มาก

---

8. วิธีคุยกับ Claude ให้ได้ผลกับ Wiki

เวลาคุยกับ Claude อย่าคิดว่าเรากำลังค้น Google

ให้คิดว่าเรากำลัง brief ผู้ช่วยคนหนึ่ง

ผู้ช่วยที่ดีต้องรู้ 5 อย่าง:

1. งานคืออะไร

2. context คืออะไร
3. source อยู่ไหน
4. output ต้องหน้าตาแบบไหน
5. อะไรคือข้อห้าม

Prompt แบบไม่ชัด:

> ช่วยสรุปให้หน่อย

Prompt แบบใช้ได้:

```
ฉันจะเอา note นี้เข้า Obsidian เพื่อใช้ใน LLM Wiki
ช่วยสรุปเป็น Markdown โดยมีหัวข้อต่อไปนี้:

# Summary
# Key facts
# Useful ideas
# Related notes
# Questions to follow up

กติกา:
- ใช้เฉพาะข้อมูลใน source นี้
- แยก fact กับ idea
- ถ้าเจอสิ่งที่ควรตรวจต่อ ให้ใส่ใน Questions
- เขียนให้คนทำงานทั่วไปอ่านรู้เรื่อง
```

ต่างกันมาก

Claude ไม่ต้องการ prompt ยาวเสมอไป

แต่ต้องการ prompt ที่บอกงานชัด

---

9. Prompt แรกสำหรับใช้กับ Wiki ของคุณ

เมื่อคุณมี vault แรกจากบทที่ 1 แล้ว ให้ลองใช้ prompt นี้กับ note จริง 1 ชิ้น

```
ฉันกำลังสร้าง LLM Wiki ใน Obsidian
นี่คือ note/source ชิ้นแรกของฉัน

ช่วยทำ 6 อย่าง:
1. สรุปเนื้อหาเป็นภาษาไทยสั้น ๆ
2. แยก fact / inference / idea
3. เสนอชื่อ note ที่เหมาะสม
4. เสนอ internal links ที่ควรสร้างใน Obsidian
5. เสนอ tag ไม่เกิน 3 tag
6. บอกคำถามที่ควรค้นต่อ

กติกา:
- อย่าแต่งข้อมูลที่ไม่มีใน source
- ถ้าไม่แน่ใจ ให้เขียนว่า "ไม่แน่ใจ"
- เขียนผลลัพธ์เป็น Markdown ที่ copy ไปวางใน Obsidian ได้ทันที
```

จากนั้น copy ผลลัพธ์ไปใส่ใน note ของคุณ

ยังไม่ต้องทำระบบใหญ่

เริ่มจาก note เดียวก่อน

---

10. ตัวอย่างสั้น: หมิวใช้ Claude กับ meeting note

หมิวมี meeting note ดิบแบบนี้:

```
ประชุมทีม sales 25 พ.ค.
ลูกค้าถามบ่อยว่า AI training ต้องรู้โค้ดไหม
หลายคนกลัวทีมใช้ไม่เป็น
ทีม sales อยากได้ FAQ หน้าเว็บ
ควรมี webinar สำหรับผู้บริหารที่ไม่รู้ technical
ต้องส่ง draft FAQ ภายในศุกร์นี้
```

หมิวส่งให้ Claude พร้อม prompt จากหัวข้อก่อนหน้า

ผลลัพธ์ที่ควรได้อาจเป็นแบบนี้:

```markdown
# คำถามลูกค้าเรื่อง AI training ต้องรู้โค้ดไหม

type: source
status: inbox
date: 2026-05-25
source: meeting ทีม sales

## Summary
ทีม sales พบว่าลูกค้าถามบ่อยว่า AI training ต้องรู้โค้ดหรือไม่ และกังวลว่าทีมจะใช้ AI ไม่เป็น จึงควรทำ FAQ หน้าเว็

## Fact
- ลูกค้าถามบ่อยว่า AI training ต้องรู้โค้ดไหม
- ลูกค้าหลายคนกลัวทีมใช้ไม่เป็น
- ทีม sales อยากได้ FAQ หน้าเว็บ
- มี deadline ส่ง draft FAQ ภายในศุกร์นี้

## Inference
- ลูกค้ากลุ่มนี้อาจเข้าใจว่า AI training ผูกกับ programming
- เนื้อหาขายควรย้ำว่าเหมาะกับคนไม่ technical

## Idea
- ทำ webinar สำหรับผู้บริหารที่ไม่รู้ technical
- ทำ FAQ หน้าเว็บเรื่อง “ต้องรู้โค้ดไหม”

## Related notes
- [[Pain point ของเจ้าของธุรกิจ SME]]
- [[FAQ หน้าเว็บคอร์ส AI]]
- [[Webinar AI สำหรับผู้บริหาร]]

## Tags
#source #sales #faq

## Questions to follow up
- ลูกค้ากลุ่มไหนถามเรื่องนี้มากที่สุด
- มีตัวอย่างคำถามจริงจากลูกค้ากี่ข้อ
- FAQ ควรตอบด้วย tone แบบไหน
```

นี่คือรูปแบบที่ Claude ช่วยได้ดีมาก

มันไม่ได้แค่สรุป แต่มันช่วยเปลี่ยนข้อมูลดิบให้กลายเป็น note ที่เอาไปใช้ต่อได้

11. Claude ช่วยได้ แต่คุณต้องเป็น editor

ให้จำบทบาทแบบนี้:

> Obsidian = บ้านของความรู้
> Claude = ผู้ช่วยอ่านและจัดบ้าน
> คุณ = เจ้าของบ้านและ editor สุดท้าย

Claude ช่วยจัดของได้

แต่คุณต้องตัดสินใจว่า:

- note นี้ถูกไหม
- source นี้น่าเชื่อไหม
- link นี้ควรมีไหม
- idea นี้ใช้ได้จริงไหม
- อะไรควรเก็บ อะไรควรลบ

ถ้าคุณปล่อยให้ Claude จัดทุกอย่างโดยไม่ตรวจ Wiki จะดูดี แต่ข้างในอาจเต็มไปด้วยความมั่วที่เขียนสวย

อย่าหลงคำตอบที่ลื่น

ให้ดู source และโครงสร้าง

---

12. Workflow ง่าย ๆ หลังจบบทนี้

หลังจากวันนี้ เวลาเจอข้อมูลใหม่ ให้ทำแบบนี้:

1. เอาข้อมูลเข้า `00-inbox`
2. ให้ Claude สรุปเป็น Markdown
3. ให้ Claude แยก fact / inference / idea
4. ให้ Claude เสนอ note title, links, tags
5. คุณอ่านตรวจ

6. ย้าย note ไป folder ที่เหมาะสม
7. link กลับไป project หรือ index

ใช้เวลาไม่ควรเกิน 10–15 นาทีต่อ source หนึ่งชิ้น

ถ้านานกว่านั้น แปลว่าคุณอาจกำลังจัดละเอียดเกินไป

สรุปท้ายบท

Claude จะช่วย LLM Wiki ได้ดีที่สุดเมื่อคุณให้มันทำงานกับ context ที่จัดไว้ดี

บทนี้มีแก่นแค่นี้:

- Claude ไม่ควรเป็นที่เก็บความรู้หลักเพียงที่เดียว
- Obsidian คือคลังที่คุณคุมเอง
- Claude ช่วยอ่าน สรุป แตก note เชื่อมโยง และถามกลับ
- Project ใช้เป็นพื้นที่ทำงานเฉพาะเรื่อง
- Files คือของที่ให้ Claude อ่าน
- Artifacts คือชิ้นงานที่ Claude ร่างให้ใช้ต่อ
- Context window คือโต๊ะทำงาน อย่าวางทุกอย่างลงไปพร้อมกัน
- Fact / inference / idea ต้องแยกกันเสมอ
- คุณยังเป็น editor สุดท้าย

บทต่อไป เราจะเอาสองอย่างนี้มารวมเป็น pattern ชัด ๆ:

LLM Wiki คืออะไร และต่างจากโน้ตธรรมดายังไง

เพราะเมื่อมี Obsidian เป็นบ้าน และ Claude เป็นผู้ช่วย ขั้นต่อไปคือการออกแบบ “ชนิดของโน้ต” ให้ทั้งคนและ AI อ่านแล้วใช้ต่อได้จริง

Artifact ท้ายบท: Prompt แรกสำหรับใช้กับ Wiki

```
ฉันกำลังสร้าง LLM Wiki ใน Obsidian
นี่คือ note/source ชิ้นแรกของฉัน

ช่วยทำ 6 อย่าง:
1. สรุปเนื้อหาเป็นภาษาไทยสั้น ๆ
2. แยก fact / inference / idea
3. เสนอชื่อ note ที่เหมาะสม
4. เสนอ internal links ที่ควรสร้างใน Obsidian
5. เสนอ tag ไม่เกิน 3 tag
6. บอกคำถามที่ควรค้นต่อ

กติกา:
- อย่าแต่งข้อมูลที่ไม่มีใน source
- ถ้าไม่แน่ใจ ให้เขียนว่า "ไม่แน่ใจ"
- เขียนผลลัพธ์เป็น Markdown ที่ copy ไปวางใน Obsidian ได้ทันที
```

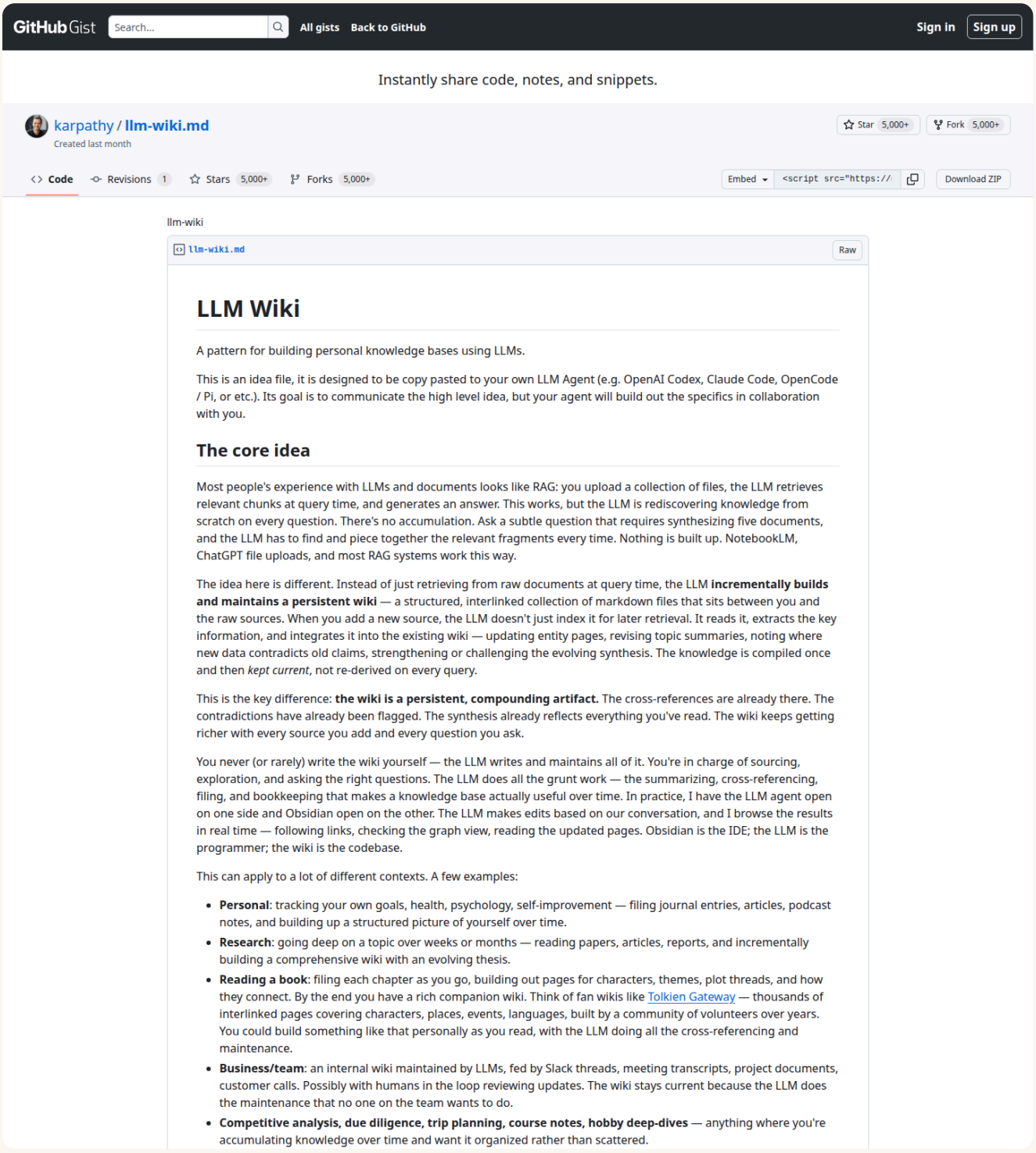
Artifact ท้ายบท: บทบาท 3 ฝ่าย

```
Obsidian = บ้านของความรู้
Claude = ผู้ช่วยอ่านและจัดบ้าน
คุณ = เจ้าของบ้านและ editor สุดท้าย
```

อ้างอิงหลักของบทนี้

- Claude Help Center: Upload files to Claude
- Claude Help Center: What are projects?
- Claude Help Center: How can I create and manage projects?
- Claude Help Center: What are artifacts and how do I use them?
- Claude Help Center: Enabling and using web search
- Claude Help Center: Using Research on Claude
- Claude Help Center: Use Claude's chat search and memory to build on previous context

LLM Wiki คืออะไร และต่างจากโน้ตธรรมดายังไง



gist เรื่อง LLM Wiki ของ Andrej Karpathy: ที่มาของแนวคิดจัดความรู้ให้ LLM ใช้ต่อได้

ภาพ: gist เรื่อง LLM Wiki ของ Andrej Karpathy: ที่มาของแนวคิดจัดความรู้ให้ LLM ใช้ต่อได้

ตอนนี้เรามีสองอย่างแล้ว

จากบทที่ 1:

> `Obsidian = บ้านของความรู้`

จากบทที่ 2:

> `Claude = ผู้ช่วยอ่านและจัดบ้าน`

บทนี้คือคำตอบของคำถามต่อไป:

> แล้วในบ้านหลังนี้ เราควรจัดของเป็นแบบไหน?

คำตอบคือ **LLM Wiki**

ไม่ต้องตกใจกับชื่อ

LLM Wiki ไม่ใช่ระบบ technical สำหรับ programmer

สำหรับคนทำงานทั่วไป ให้เข้าใจแบบนี้:

> LLM Wiki คือคลังความรู้ที่เขียนให้ทั้งคนและ AI อ่านรู้เรื่อง

มันต่างจากโน้ตธรรมดาตรงที่ไม่ได้เก็บแค่ “สิ่งที่เราเคยจด”

แต่มันจัดความรู้ให้ตอบคำถามได้ เช่น:

- เรื่องนี้มาจาก source ไหน
- fact คืออะไร
- ข้อสรุปของเราคืออะไร
- เกี่ยวกับโปรเจกต์ไหน
- มี decision อะไรเกิดขึ้นแล้ว

- ถ้าจะให้ Claude ช่วยต่อ ต้องอ่าน note ไหนก่อน

พูดสั้น ๆ:

โน้ตธรรมดาเก็บข้อมูล

LLM Wiki เก็บข้อมูล + ความสัมพันธ์ + ที่มา + วิธีใช้ต่อ

---

1. ปัญหาของโน้ตธรรมดา

คนทำงานจำนวนมากไม่ได้ขาดข้อมูล

เรามีข้อมูลเยอะเกินไปด้วยซ้ำ

ปัญหาคือข้อมูลอยู่แบบนี้:

> ไฟล์ meeting อยู่ที่หนึ่ง
> ไอเดียอยู่ใน chat
> ลิงก์อยู่ใน browser
> เอกสารลูกค้าอยู่ใน drive
> สรุปที่ Claude เคยทำอยู่ในบทสนทนาเก่า
> decision อยู่ในหัวใครบางคน

ต่อให้คุณเอาทุกอย่างมาใส่ Obsidian ถ้าไม่มีโครง มันก็แค่ย้ายความรกจากหลายที่มาอยู่ที่เดียว

LLM Wiki จึงไม่ใช่แค่ “เก็บทุกอย่าง”

แต่คือการทำให้ความรู้แต่ละชิ้นมีหน้าที่ชัด

บาง note เป็น source

บาง note เป็น concept

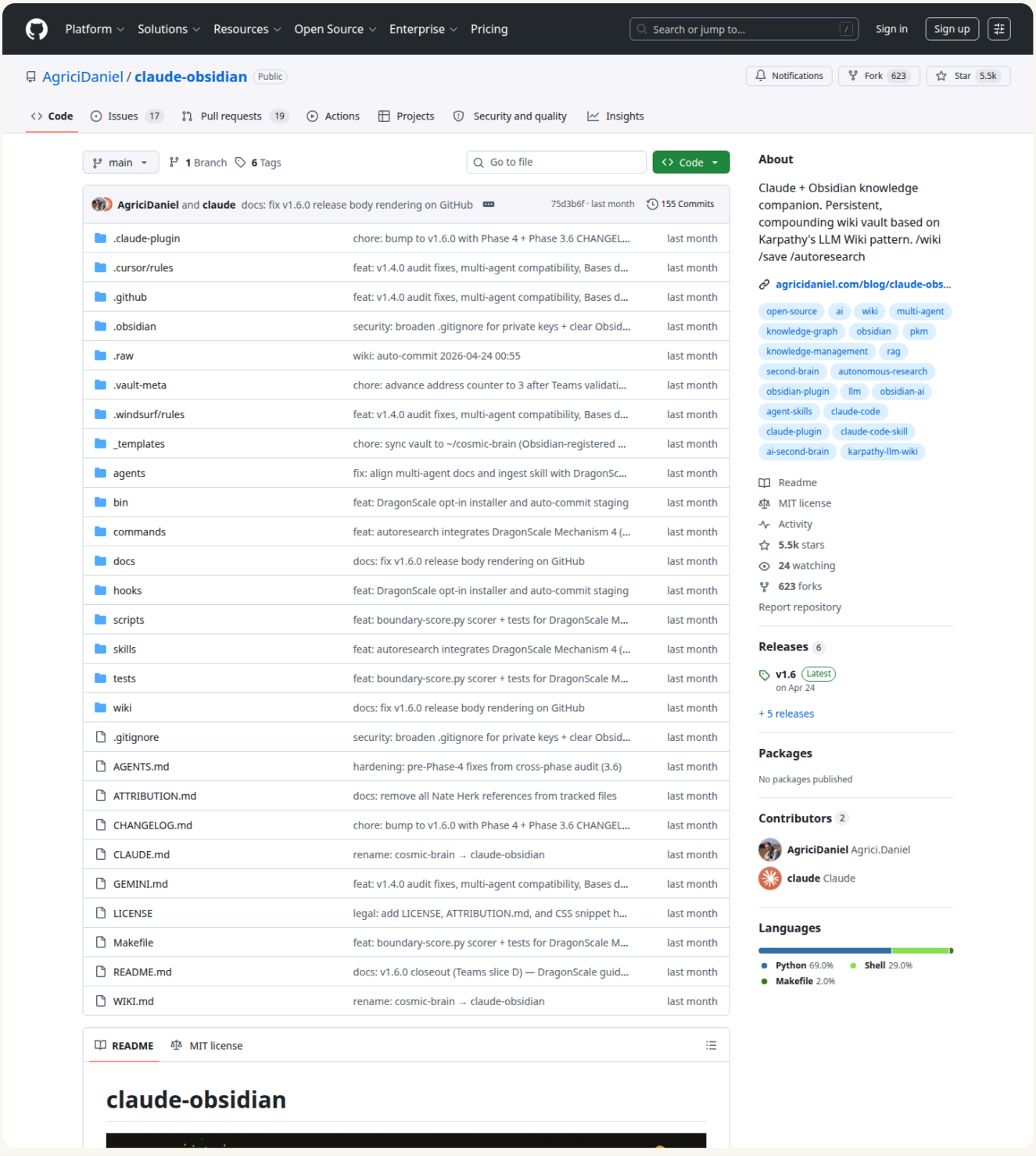
บาง note เป็น project

บาง note เป็น decision

บาง note เป็น index

พอแยกแบบนี้ คุณจะหาเจอง่ายขึ้น และ Claude จะช่วยต่อได้แม่นขึ้น

2. แก่นของ LLM Wiki ในเล่มนี้



GitHub repo claude-obsidian: ตัวอย่าง LLM Wiki ที่ใช้ Claude กับ Obsidian จริง

ภาพ: GitHub repo claude-obsidian: ตัวอย่าง LLM Wiki ที่ใช้ Claude กับ Obsidian จริง

สำหรับเล่มนี้ เราจะใช้ LLM Wiki แบบเรียบง่ายที่สุด

ไม่ต้องมีระบบค้นหาใหญ่

ไม่ต้องมีฐานข้อมูล

ไม่ต้องมี plugin เยอะ

ใช้แค่ Markdown note + link + source + index ก็เริ่มได้แล้ว

แก่นมี 5 ข้อ:

1. **Raw source ต้องอยู่ได้**: สิ่งที่เป็นต้นทางควรเก็บไว้หรืออ้างกลับได้
2. **Wiki note ต้องอ่านง่าย**: สรุปแล้ว แยกหัวข้อแล้ว ใช้ต่อได้
3. **Link ต้องช่วยเดินทาง**: note สำคัญควรเชื่อมกัน
4. **Index ต้องพาไปถูกที่**: ไม่ต้องเปิดทุก folder เอง
5. **Claude ต้องรู้บทบาท**: ช่วยจัดและสรุป แต่ไม่แต่ง fact

ถ้าทำได้ 5 ข้อนี้ คุณมี LLM Wiki แบบใช้งานจริงแล้ว

ยังไม่ต้องสนใจคำว่า RAG, vector, embedding หรือ automation

---

3. Raw source กับ Wiki note ต้องแยกกัน

นี่คือกติกาที่สำคัญที่สุดข้อหนึ่ง

Source คือของต้นทาง เช่น:

- บทความ
- PDF
- meeting transcript
- email จากลูกค้า
- หน้าเว็บ official
- research note
- ข้อความสัมภาษณ์

Wiki note คือสิ่งที่เราหรือ Claude สรุปและจัดระเบียบจาก source นั้น

ทำไมต้องแยก?

เพราะ source คือหลักฐาน

ส่วน wiki note คือความเข้าใจที่อาจถูกปรับได้

ตัวอย่าง:

```
source: transcript ประชุมทีม sales
wiki note: สรุป pain point ลูกค้าเรื่อง AI training
```

ถ้าวันหนึ่ง wiki note สรุปผิด คุณยังกลับไปดู source ได้

ถ้าวันหนึ่ง Claude เขียนสวยแต่มั่ว คุณยังมีต้นทางให้ตรวจ

ถ้าวันหนึ่งมีข้อมูลใหม่ คุณอัปเดต wiki note ได้โดยไม่ทำลาย source เดิม

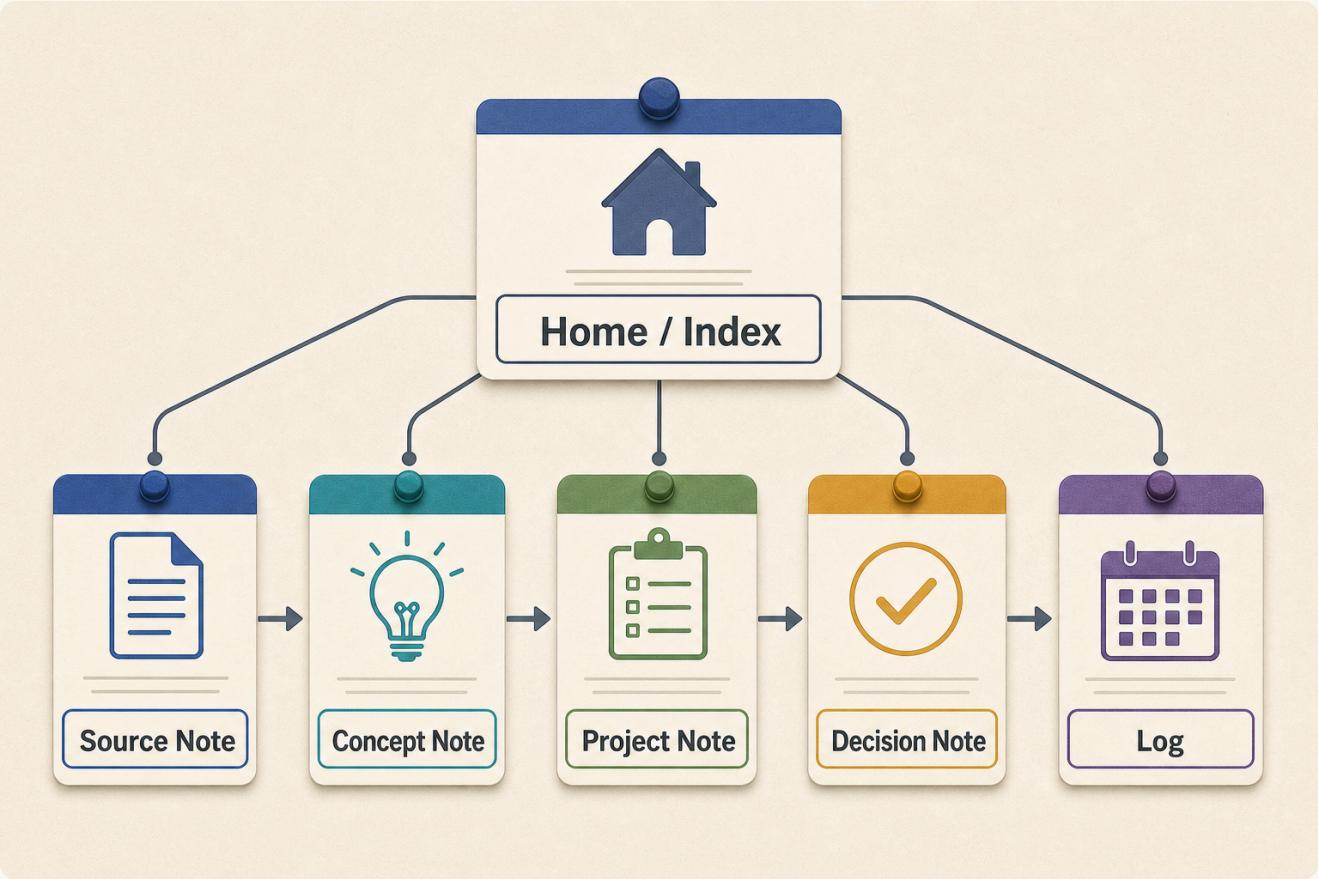
นี่คือเหตุผลที่เราพูดคำว่า **source-first** ตลอดเล่ม

ไม่ใช่เพราะอยากให้ดูเป็นทางการ

แต่เพราะมันกัน wiki กลายเป็นกองคำตอบสวย ๆ ที่ไม่รู้ว่ามาจากไหน

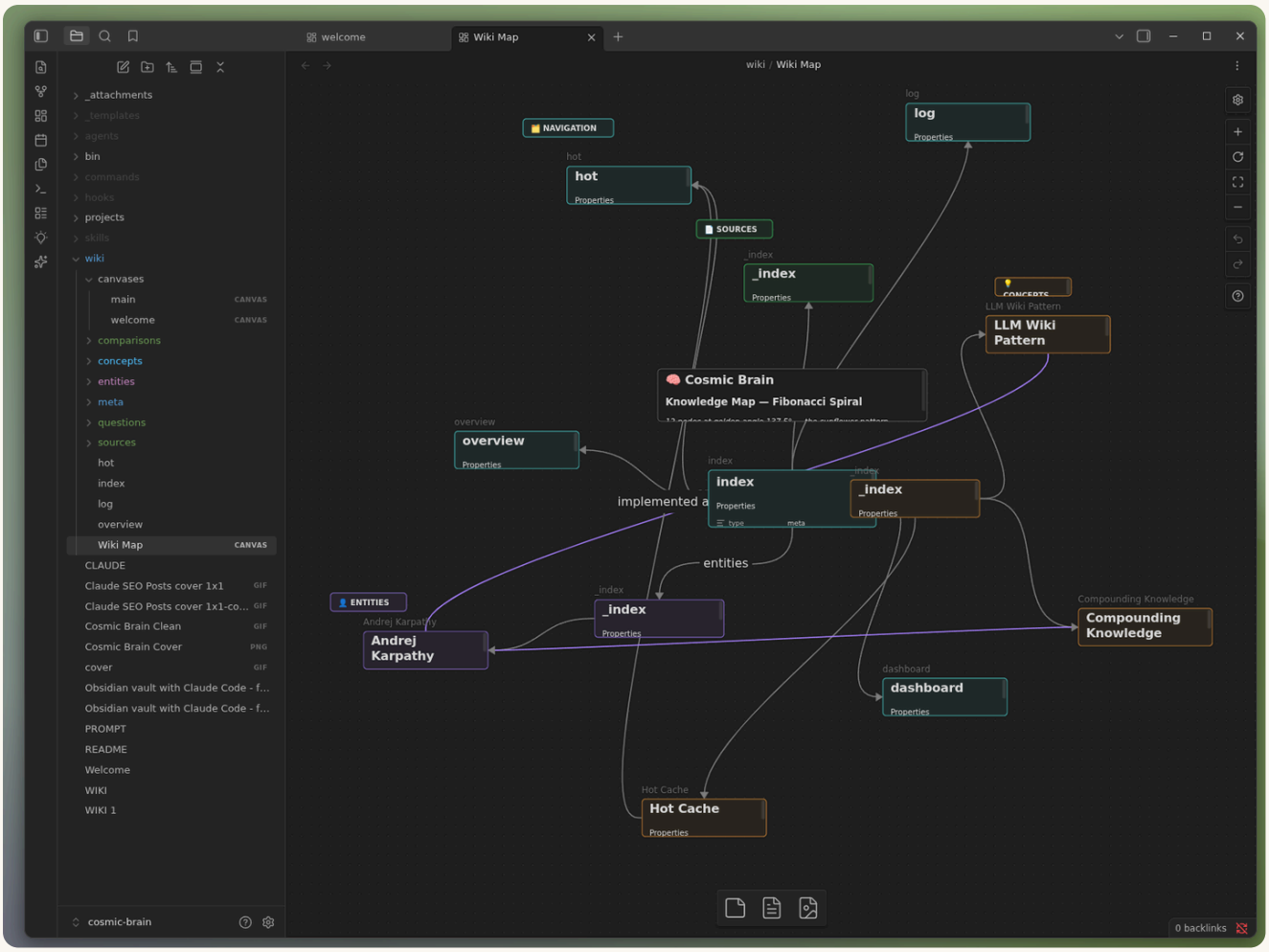
---

4. ชนิดของ note ที่ควรมี



LLM Wiki เริ่มจาก note ไม่กี่ชนิด แต่เชื่อมกันให้คนและ AI อ่านต่อได้ง่าย

ภาพ: LLM Wiki เริ่มจาก note ไม่กี่ชนิด แต่เชื่อมกันให้คนและ AI อ่านต่อได้ง่าย



ตัวอย่าง Wiki Map: ใช้ให้เห็นภาพความเชื่อมโยง ไม่ใช่สิ่งที่มือใหม่ต้องทำตั้งแต่วันแรก

ภาพ: ตัวอย่าง Wiki Map: ใช้ให้เห็นภาพความเชื่อมโยง ไม่ใช่สิ่งที่มือใหม่ต้องทำตั้งแต่วันแรก

สำหรับคนเริ่มต้น ใช้ note แค่ 5 ชนิดพอ

> 1. Source note
> 2. Concept note
> 3. Project note
> 4. Decision note
> 5. Index/Home note

ไม่ต้องมีมากกว่านี้ในช่วงแรก

ถ้าคุณเริ่มจากชนิด note 15 แบบ คุณจะเสียเวลาคิดมากกว่าทำงาน

เราจะดูทีละชนิด

---

5. Source note: โน้ตที่บอกว่า “ข้อมูลนี้มาจากไหน”

Source note ใช้เก็บข้อมูลจากแหล่งต้นทางหนึ่งชิ้น

หนึ่งบทความ = หนึ่ง source note

หนึ่ง meeting = หนึ่ง source note

หนึ่ง PDF = หนึ่ง source note

หนึ่งหน้าเว็บสำคัญ = หนึ่ง source note

Source note ควรตอบคำถามเหล่านี้:

- source นี้คืออะไร
- มาจากไหน
- มี fact สำคัญอะไร
- เกี่ยวกับงานเราอย่างไร
- ควรแตกเป็น concept note อะไรบ้าง

Template เริ่มต้น:

MARKDOWN

```
# ชื่อ source

type: source
status: inbox
date: YYYY-MM-DD
source: ใส่ลิงก์หรือที่มา

## Summary
สรุปสั้น ๆ 3–5 บรรทัด

## Key facts
-
-
-

## Useful for
- ใช้กับงาน/โปรเจกต์ไหน

## Related notes
- [[ชื่อ note ที่เกี่ยวข้อง]]

## Questions
- คำถามที่ควรค้นต่อ
```

ตัวอย่าง:

```markdown
# Meeting ทีม sales เรื่องคำถามลูกค้า AI training

type: source
status: reviewed
date: 2026-05-25
source: meeting ทีม sales

## Summary
ทีม sales พบว่าลูกค้าถามบ่อยว่า AI training ต้องรู้โค้ดไหม และกังวลว่าทีมจะใช้ AI ไม่เป็น ทีมจึงอยากได้ FAQ หน้า

## Key facts
- ลูกค้าถามบ่อยว่า AI training ต้องรู้โค้ดไหม
- ลูกค้ากังวลว่าทีมจะใช้ AI ไม่เป็น
- ทีม sales ขอ FAQ สำหรับหน้าเว็บ
- ต้องส่ง draft FAQ ภายในศุกร์นี้

## Useful for
- [[FAQ หน้าเว็บคอร์ส AI]]
- [[AI training สำหรับคนไม่รู้โค้ด]]

## Related notes
- [[Pain point ของเจ้าของธุรกิจ SME]]
- [[AI training สำหรับคนไม่รู้โค้ด]]

## Questions
- ลูกค้ากลุ่มไหนถามเรื่องนี้มากที่สุด
- มีคำถามจริงจากลูกค้าอีกกี่ข้อ
```

Source note ทำให้ Claude รู้ว่าอะไรคือข้อมูลต้นทาง

เวลาคุณถาม Claude ในอนาคต มันจะไม่ต้องเดาว่าเรื่องนี้มาจากไหน

---

6. Concept note: โน้ตที่อธิบายแนวคิดหนึ่งเรื่อง

Concept note คือโน้ตที่อธิบาย “เรื่อง” หรือ “แนวคิด” ที่ใช้ซ้ำได้

ตัวอย่าง:

```
Pain point ของเจ้าของธุรกิจ SME
AI training สำหรับคนไม่รู้โค้ด
Source-first thinking
Weekly review
Customer objection
```

Concept note ต่างจาก source note ตรงนี้:

- Source note ผูกกับแหล่งข้อมูลหนึ่งชิ้น
- Concept note รวมความเข้าใจจากหลาย source

Concept note ควรตอบ:

- เรื่องนี้คืออะไร
- ทำไมสำคัญ
- มีตัวอย่างอะไร
- ใช้กับงานไหน
- มี source ไหนรองรับ

Template:

```markdown
# ชื่อ concept

type: concept
status: draft

## ความหมายสั้น ๆ
อธิบายด้วยภาษาคนทั่วไป

## ทำไมเรื่องนี้สำคัญ
-

## ตัวอย่าง
-

## ใช้กับงานไหน
- [[ชื่อ project]]

## Sources
- [[ชื่อ source note]]

## Related concepts
- [[ชื่อ concept อื่น]]
```

ตัวอย่าง:

MARKDOWN

```
# AI training สำหรับคนไม่รู้โค้ด

type: concept
status: draft

## ความหมายสั้น ๆ
การสอนใช้ AI เพื่อช่วยงานธุรกิจ โดยไม่ตั้งต้นจากการเขียนโปรแกรม แต่ตั้งต้นจากงานจริง เช่น marketing, sales, op

## ทำไมเรื่องนี้สำคัญ
- ลูกค้าหลายคนเข้าใจว่า AI training ต้องเกี่ยวกับ programming
- ถ้าสื่อสารไม่ชัด ลูกค้าอาจคิดว่าคอร์สไม่เหมาะกับทีมของเขา

## ตัวอย่าง
- ใช้ Claude สรุป meeting
- ใช้ Claude ทำ FAQ จากคำถามลูกค้า
- ใช้ Claude ช่วยร่าง FAQ หรือ outline webinar

## ใช้กับงานไหน
- [[FAQ หน้าเว็บคอร์ส AI]]
- [[Webinar AI สำหรับผู้บริหาร]] ถ้าทีมตัดสินใจทำจริง

## Sources
- [[Meeting ทีม sales เรื่องคำถามลูกค้า AI training]]

## Related concepts
- [[Pain point ของเจ้าของธุรกิจ SME]]
- [[ผู้บริหารไม่รู้จะเริ่มใช้ AI ตรงไหน]]
```

Concept note คือสิ่งที่ทำให้ wiki เริ่ม “คิดได้”

เพราะมันไม่ใช่แค่เก็บข้อมูล แต่เริ่มรวม pattern จากหลาย source

---

7. Project note: ศูนย์กลางของงานหนึ่งชิ้น

Project note คือหน้าเดียวที่รวม context ของงานหนึ่งงาน

เช่น:

> Webinar AI เดือนมิถุนายน
> ปรับหน้าเว็บคอร์ส AI
> ทำ knowledge base ทีม sales
> เขียนหนังสือ Claude + Obsidian

Project note ช่วยให้คุณไม่ต้องถามว่า “ไฟล์ของโปรเจกต์นี้อยู่ไหนบ้าง?”

ทุกอย่างควรกลับมาที่ project note

Template:

```markdown
# ชื่อโปรเจกต์

type: project
status: active
owner:
deadline:

## เป้าหมาย
โปรเจกต์นี้ต้องการผลลัพธ์อะไร

## Context สำคัญ
-

## Sources
- [[source note]]

## Concepts ที่เกี่ยวข้อง
- [[concept note]]

## Decisions
- [[decision note]]

## Todo
- [ ]
- [ ]

## Next review
YYYY-MM-DD
```

ตัวอย่าง:

```markdown
# FAQ หน้าเว็บคอร์ส AI

type: project
status: active
owner: หมิว
deadline: ศุกร์นี้

## เป้าหมาย
สร้าง FAQ หน้าเว็บเพื่อตอบข้อกังวลของลูกค้าว่า AI training ต้องรู้โค้ดไหม และทีม business ใช้ได้จริงหรือไม่

## Context สำคัญ
- ลูกค้าถามบ่อยว่า AI training ต้องรู้โค้ดไหม
- ลูกค้าหลายคนกังวลว่าทีมจะใช้ AI ไม่เป็น
- ทีม sales ต้องการ FAQ สำหรับหน้าเว็บ

## Sources
- [[Meeting ทีม sales เรื่องคำถามลูกค้า AI training]]

## Concepts ที่เกี่ยวข้อง
- [[AI training สำหรับคนไม่รู้โค้ด]]
- [[Pain point ลูกค้าเรื่องทีมใช้ AI ไม่เป็น]]

## Decisions
- ยังไม่มี decision ชัดเจนใน source นี้

## Todo
- [ ] ขอคำถามจริงจากทีม sales
- [ ] ร่าง FAQ 10 ข้อ
- [ ] ส่ง draft ให้ทีม sales ตรวจ

## Next review
YYYY-MM-DD
```

เวลาให้ Claude ช่วยงาน project นี้ คุณไม่ต้องเล่าทุกอย่างใหม่

ส่ง project note พร้อม note ที่ link อยู่ แล้วให้ Claude สรุป context กลับมาก่อนเริ่มทำงาน

---

8. Decision note: โน้ตที่บอกว่า “เราตัดสินใจอะไรไปแล้ว”

นี่คือ note ที่หลายคนไม่มี แต่ควรมีมาก

เพราะงานจริงไม่ได้พังเพราะไม่มีไอเดีย

งานจริงพังเพราะลืมว่าเคยตัดสินใจอะไรไปแล้ว

Decision note ใช้เก็บการตัดสินใจสำคัญ เช่น:

- เลือกกลุ่มเป้าหมายไหนก่อน
- ใช้ชื่อ campaign ว่าอะไร
- ตัด feature ไหนออก
- เลื่อน deadline เพราะอะไร
- ทำไมไม่เลือกทางเลือกหนึ่ง

Template:

```markdown
# ชื่อ decision

type: decision
date: YYYY-MM-DD
status: decided

## Decision
เราตัดสินใจว่าอะไร

## Why
เหตุผลคืออะไร

## Options considered
- ทางเลือก A
- ทางเลือก B

## Evidence / Sources
- [[source note]]

## Impact
สิ่งที่ต้องทำต่อหรือสิ่งที่จะเปลี่ยน

## Related project
- [[project note]]
```

ตัวอย่างด้านล่างใช้ในกรณีที่ทีม “ยืนยันแล้ว” ว่าจะทำ webinar จริง

ถ้ายังเป็นแค่คำว่า “ควรมี webinar” ให้เก็บไว้เป็น Idea ก่อน อย่าเพิ่งสร้าง decision note

```markdown
# Decision - ทำ webinar สำหรับผู้บริหารที่ไม่ technical - 2026-05-25

type: decision
date: 2026-05-25
status: decided

## Decision
ทีมตัดสินใจทำ webinar หัวข้อ AI สำหรับผู้บริหารที่ไม่รู้โค้ด

## Why
ทีมต้องการตอบข้อกังวลของลูกค้าว่า AI training ต้องรู้โค้ดไหม และทีม business ใช้ได้จริงหรือไม่

## Options considered
- ทำเฉพาะ FAQ หน้าเว็บ
- ทำ webinar สำหรับผู้บริหารเพิ่มเติม

## Evidence / Sources
- [[Meeting ทีม sales เรื่องคำถามลูกค้า AI training]]
- [[Meeting ที่ทีมยืนยันแผน webinar]]

## Impact
- ต้องสร้าง project note สำหรับ webinar
- หน้า landing page ต้องย้ำว่าไม่ต้องรู้โค้ด
- ตัวอย่างใน webinar ต้องเป็นงาน business จริง

## Related project
- [[Webinar AI สำหรับผู้บริหาร]]
```

Decision note ช่วย Claude มาก เพราะมันกันไม่ให้ Claude พูดเหมือน idea เป็นสิ่งที่ตกลงแล้ว

---

9. Index note: ประตูหน้าบ้านของ Wiki

Index note คือหน้าแผนที่

ถ้า wiki มี note เยอะขึ้น แต่ไม่มี index คุณจะเริ่มหลง

Claude ก็หลงได้เหมือนกัน

Index note ไม่ต้องซับซ้อน

เริ่มจากหน้า `Home` หรือ `Index` แบบนี้:

```markdown
# Home

## Projects สำคัญ
- [[FAQ หน้าเว็บคอร์ส AI]]

## Concepts สำคัญ
- [[AI training สำหรับคนไม่รู้โค้ด]]
- [[Pain point ของเจ้าของธุรกิจ SME]]

## Sources ล่าสุด
- [[Meeting ทีม sales เรื่องคำถามลูกค้า AI training]]

## Decisions ล่าสุด
- ยังไม่มี decision ชัดเจนจาก source แรก

## ต้อง review
- [[00-inbox]]
```

ให้คิดว่า index note คือหน้าแรกที่คุณและ Claude ควรอ่านก่อนเริ่มงาน

ถ้า project ใหญ่ขึ้น อาจมีหลาย index เช่น:

- `Index ลูกค้า`
- `Index content ideas`
- `Index โปรเจกต์ active`
- `Index sources`

แต่เริ่มจาก Home หน้าเดียวก่อนพอ

---

10. Wiki ที่ดีต้องมี log แบบง่าย

ถ้าคุณอยากเพิ่มอีกหนึ่งไฟล์ ให้เพิ่ม `Log.md`

Log คือบันทึกว่า wiki เปลี่ยนอะไรไปบ้าง

ไม่ต้องละเอียดมาก

ตัวอย่าง:

```markdown
# Log

## 2026-05-25
- เพิ่ม source note จาก meeting ทีม sales
- แตก concept note: [[AI training สำหรับคนไม่รู้โค้ด]]
- สร้าง project note: [[FAQ หน้าเว็บคอร์ส AI]]
- บันทึก idea: webinar สำหรับผู้บริหารที่ไม่ technical
```

Log มีประโยชน์เมื่อคุณกลับมาหลังจากหายไปหลายวัน แล้วจำไม่ได้ว่าทำอะไรไว้

และมีประโยชน์มากเมื่อให้ Claude ช่วยต่อ

คุณสามารถส่ง `Home` + `Log` + project note ให้ Claude แล้วถามว่า:

> ช่วยสรุปสถานะล่าสุดของ `wiki` นี้ และบอกว่าควรทำอะไรต่อ

---

11. LLM Wiki ต่างจาก RAG ยังไง แบบไม่ technical

คุณอาจเคยได้ยินคำว่า RAG

ไม่จำเป็นต้องเข้าใจลึก

ให้จำง่าย ๆ แบบนี้:

> `RAG = ตอนถาม ค่อยไปค้นเอกสารมาประกอบคำตอบ`
> `LLM Wiki = อ่านและจัดความรู้ไว้ก่อน แล้วค่อยถามจากสิ่งที่จัดแล้ว`

เปรียบเทียบแบบชีวิตจริง:

RAG เหมือนคุณมีกล่องเอกสารใหญ่ ๆ แล้วทุกครั้งที่มีคำถาม ผู้ช่วยต้องรื้อกล่องใหม่

LLM Wiki เหมือนคุณมีแฟ้มที่จัดหมวดแล้ว มีสารบัญ มีสรุป มี post-it บอกว่าเรื่องไหนเกี่ยวกับเรื่องไหน

สำหรับคนทำงานทั่วไปที่มี source หลักสิบถึงหลักร้อย LLM Wiki มักเริ่มง่ายกว่า เพราะใช้แค่ note + link + index

แต่ไม่ต้องเอาไปเถียงว่าอะไรดีกว่าทุกกรณี

เล่มนี้สนใจแค่คำถามเดียว:

> วิธีไหนทำให้คุณใช้ Claude กับความรู้ของตัวเองได้ดีขึ้นเร็วที่สุด?

สำหรับเป้าหมายนี้ LLM Wiki แบบเรียบง่ายตอบโจทย์

---

12. กติกา 6 ข้อของ LLM Wiki ขนาดเล็ก

ถ้าจำทั้งบทไม่ได้ ให้จำ 6 ข้อนี้

1. Source ไม่ใช่ Summary

อย่าให้สรุปของ Claude แทนที่ source ทั้งหมด

ถ้ามี source สำคัญ ให้เก็บที่มาไว้เสมอ

2. หนึ่ง note หนึ่งหน้าที่

อย่าเอา source, concept, todo, decision และ project มาปนกันหมดในหน้าเดียว

3. Link เฉพาะที่ช่วยให้คิดต่อ

ไม่ต้อง link ทุกคำ

link เฉพาะเรื่องที่ควรเปิดอ่านต่อจริง ๆ

4. Index ต้องมีชีวิต

ถ้าสร้าง note สำคัญ ให้เพิ่มเข้า Home หรือ index ที่เกี่ยวข้อง

5. Claude ช่วยจัด แต่คุณตรวจ

ให้ Claude เสนอได้ แต่อย่าให้ Claude ตัดสินใจแทนทุกอย่าง

6. Wiki ต้องเล็กพอให้ใช้ต่อ

ถ้าระบบเริ่มซับซ้อนจนคุณไม่อยากเปิด แปลว่าระบบใหญ่เกินไป

ตัดทิ้งได้

---

13. Prompt สำหรับให้ Claude สร้างชุด note จาก source หนึ่งชิ้น

เมื่อคุณมี source หนึ่งชิ้น ลองใช้ prompt นี้

```
ฉันกำลังทำ LLM Wiki ใน Obsidian
ช่วยแปลง source นี้เป็นชุด note ที่ใช้ต่อได้

ขอผลลัพธ์เป็น 5 ส่วน:
1. Source note: สรุป source และ key facts
2. Concept notes ที่ควรแตกออกมา
3. Project note ที่เกี่ยวข้อง ถ้ามี
4. Decision note ถ้ามีการตัดสินใจใน source
5. Index update: ควรเพิ่ม link อะไรเข้า Home หรือ Index

กติกา:
- แยก fact / inference / idea ให้ชัด
- อย่าแต่งข้อมูลที่ไม่มีใน source
- ถ้าไม่ควรสร้าง note ใหม่ ให้บอกว่าไม่ควรสร้าง
- ใช้ชื่อ note ที่คนทำงานทั่วไปอ่านแล้วเข้าใจ
- เขียนเป็น Markdown ที่ copy ไปใส่ Obsidian ได้
```

อย่าเพิ่งใช้กับ source 20 ชิ้น

เริ่มจาก source เดียวก่อน

อ่านผลลัพธ์ ตรวจ แก้ แล้วค่อยเก็บ

---

14. ตัวอย่างจาก meeting note เดิม

จากบทที่ 2 เรามี meeting note นี้:

```
ประชุมทีม sales 25 พ.ค.
ลูกค้าถามบ่อยว่า AI training ต้องรู้โค้ดไหม
หลายคนกลัวทีมใช้ไม่เป็น
ทีม sales อยากได้ FAQ หน้าเว็บ
ควรมี webinar สำหรับผู้บริหารที่ไม่รู้ technical
ต้องส่ง draft FAQ ภายในศุกร์นี้
```

ถ้าแปลงเป็น LLM Wiki จริง ๆ เราอาจได้ 3–4 note ก่อน:

```
/10-sources/Meeting ทีม sales เรื่องคำถามลูกค้า AI training.md
/30-concepts/AI training สำหรับคนไม่รู้โค้ด.md
/30-concepts/Pain point ลูกค้าเรื่องทีมใช้ AI ไม่เป็น.md
/40-projects/FAQ หน้าเว็บคอร์ส AI.md
```

ส่วน webinar ยังเป็น idea จนกว่าทีมจะยืนยันจริง

จาก meeting note ดิบ 6 บรรทัด เราไม่ได้แค่ “สรุป”

เราเปลี่ยนมันเป็นโครงความรู้ที่ใช้ต่อได้หลายทาง:

- ทีม sales ใช้ทำ FAQ
- ทีม marketing ใช้ต่อยอดเป็น landing page
- Claude ใช้อ่าน context ก่อนร่าง FAQ หรือ content
- ทีมคุยต่อได้ว่า webinar เป็นแค่ idea หรือจะตัดสินใจทำจริง

นี่คือความต่างระหว่างโน้ตธรรมดากับ LLM Wiki

---

15. อย่าทำให้ Wiki ใหญ่เกินชีวิตจริง

ข้อผิดพลาดที่พบบ่อยคือเริ่มสนุกกับระบบ

สร้าง folder เพิ่ม

สร้าง property เพิ่ม

สร้าง template เพิ่ม

ติด plugin เพิ่ม

สุดท้ายไม่ได้ใช้ทำงานจริง

LLM Wiki ที่ดีไม่ใช่ wiki ที่ซับซ้อนที่สุด

แต่คือ wiki ที่คุณกลับมาใช้ได้ทุกสัปดาห์

กติกาสำหรับเล่มนี้คือ:

> ถ้าคุณอธิบายระบบของตัวเองให้เพื่อนร่วมงานเข้าใจไม่ได้ใน 2 นาที ระบบนั้นซับซ้อนเกินไป

ระบบเริ่มต้นของเราจึงมีแค่นี้:

```
Source note = ข้อมูลมาจากไหน
Concept note = เรื่องนี้คืออะไร
Project note = งานนี้กำลังทำอะไร
Decision note = ตัดสินใจอะไรไปแล้ว
Index note = เริ่มอ่านตรงไหน
```

พอใช้คล่องแล้วค่อยเพิ่ม

สรุปท้ายบท

LLM Wiki ไม่ใช่การจดโน้ตให้เยอะขึ้น

แต่คือการจัดความรู้ให้ใช้ต่อได้ง่ายขึ้น

บทนี้มีแก่นสำคัญ:

- Raw source คือหลักฐาน
- Wiki note คือความเข้าใจที่จัดระเบียบแล้ว
- Source note เก็บข้อมูลจากแหล่งต้นทาง
- Concept note รวมแนวคิดที่ใช้ซ้ำ
- Project note เป็นศูนย์กลางของงาน

- Decision note กันลืมว่าเคยเลือกอะไรไปแล้ว
- Index note คือประตูหน้าบ้าน
- Log ช่วยบอกว่า wiki เปลี่ยนอะไรไปบ้าง
- Claude ช่วยสร้างและดูแล note ได้ แต่คุณต้องตรวจ

บทต่อไป เราจะลงมือจัดบ้านจริง

ตั้งค่า Vault แรกใน 30 นาที

ไม่ใช่แค่รู้ว่า note มีกี่ชนิด แต่จะสร้าง folder, Home, template และ note แรกให้พร้อมใช้

Artifact ท้ายบท: Note 5 ชนิดที่ต้องรู้

```
Source note   = ข้อมูลนี้มาจากไหน
Concept note  = เรื่องนี้คืออะไร
Project note  = งานนี้กำลังทำอะไร
Decision note = ตัดสินใจอะไรไปแล้ว
Index note    = เริ่มอ่านตรงไหน
```

Artifact ท้ายบท: Prompt สร้างชุด note จาก source หนึ่งชิ้น

```
ฉันกำลังทำ LLM Wiki ใน Obsidian
ช่วยแปลง source นี้เป็นชุด note ที่ใช้ต่อได้

ขอผลลัพธ์เป็น 5 ส่วน:
1. Source note: สรุป source และ key facts
2. Concept notes ที่ควรแตกออกมา
3. Project note ที่เกี่ยวข้อง ถ้ามี
4. Decision note ถ้ามีการตัดสินใจใน source
5. Index update: ควรเพิ่ม link อะไรเข้า Home หรือ Index

กติกา:
- แยก fact / inference / idea ให้ชัด
- อย่าแต่งข้อมูลที่ไม่มีใน source
- ถ้าไม่ควรสร้าง note ใหม่ ให้บอกว่าไม่ควรสร้าง
- ใช้ชื่อ note ที่คนทำงานทั่วไปอ่านแล้วเข้าใจ
- เขียนเป็น Markdown ที่ copy ไปใส่ Obsidian ได้
```

อ้างอิงหลักของบทนี้

- Andrej Karpathy: LLM Wiki gist
- `AgriciDaniel/claude-obsidian`: LLM Wiki Pattern note
- `AgriciDaniel/claude-obsidian`: Wiki vs RAG note
- `AgriciDaniel/claude-obsidian`: Source-First Synthesis note
- `AgriciDaniel/claude-obsidian`: source/concept/question/comparison templates

บทที่ 4

ตั้งค่า Vault แรกใน 30 นาที

บทนี้เป็นบทลงมือทำ

ไม่ต้องอ่านทฤษฎีเพิ่ม

เป้าหมายคือจบบทนี้แล้วคุณมี vault แรกที่ใช้ได้จริง ไม่ใช่แค่เข้าใจว่า Obsidian คืออะไร

สิ่งที่เราจะสร้าง:

```
my-wiki
  00-inbox
  10-sources
  20-notes
  30-concepts
  40-projects
  90-index
  Home.md
  Log.md
```

และจะมี template ง่าย ๆ สำหรับ note 4 แบบ:

- Source note
- Concept note
- Project note
- Decision note

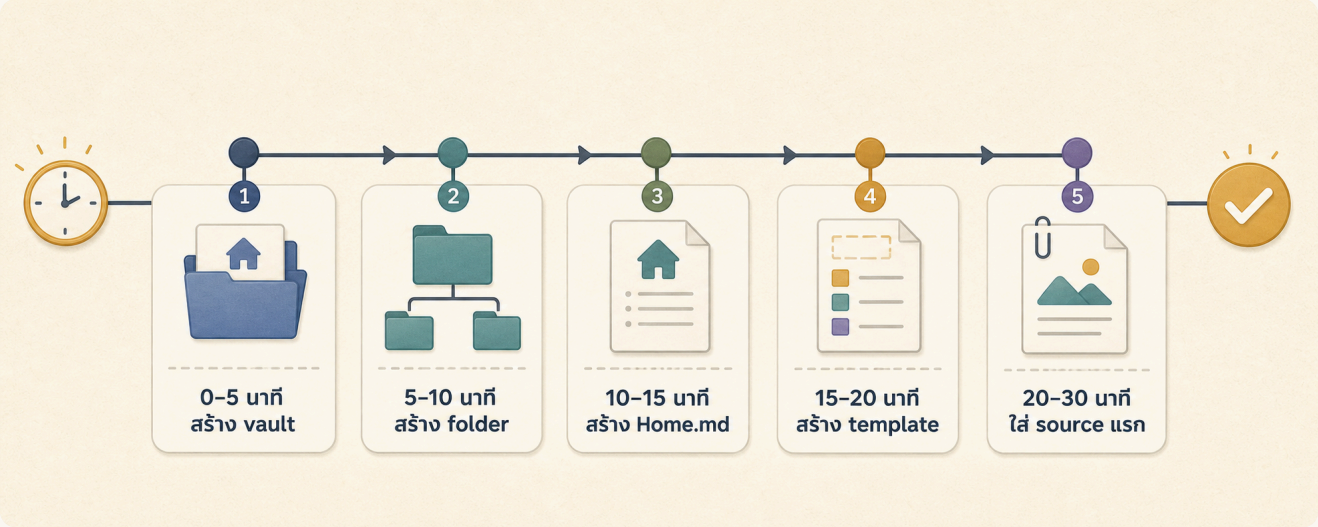
ไม่ต้องติด plugin

ไม่ต้องแต่ง theme

ไม่ต้องทำ graph ให้สวย

วันนี้เราต้องการแค่ “ระบบขั้นต่ำที่ใช้ได้จริง”

ก่อนเริ่ม: ตั้ง expectation ให้ถูก



Vault แรกไม่ต้องสมบูรณ์ แค่ทำ 5 อย่างนี้ให้เสร็จใน 30 นาทีก็เริ่มใช้ได้

ภาพ: Vault แรกไม่ต้องสมบูรณ์ แค่ทำ 5 อย่างนี้ให้เสร็จใน 30 นาทีก็เริ่มใช้ได้

Vault แรกไม่ต้องสมบูรณ์

มันเหมือนโต๊ะทำงานวันแรก

คุณยังไม่รู้ว่าชั้นไหนควรวางอะไร จนกว่าจะเริ่มทำงานจริง

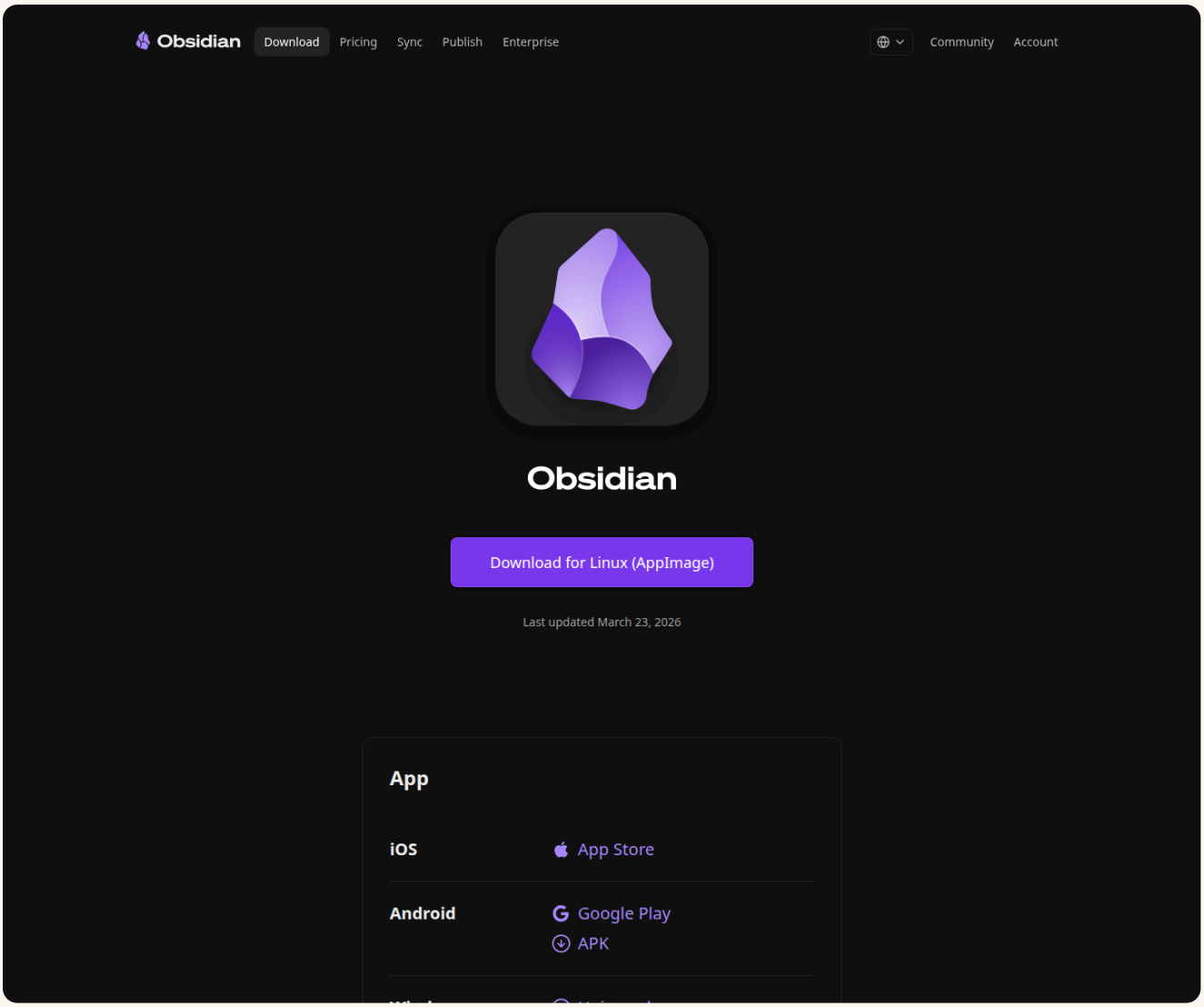
ดังนั้นอย่าเสียเวลาออกแบบมากเกินไป

กติกาของบทนี้:

1. ทำตามก่อน ปรับทีหลัง
2. เริ่มจาก folder น้อย ๆ
3. ใช้ template สั้น ๆ
4. ใส่ข้อมูลจริงอย่างน้อย 1 ชิ้น
5. ให้ Claude ช่วยจัด แต่คุณต้องอ่านตรวจ

ถ้าทำครบ 5 ข้อนี้ คุณเริ่มมี LLM Wiki แล้ว

Step 1: สร้าง vault ใหม่



หน้า Download ของ Obsidian: ขั้นแรกคือมี vault เล็ก ๆ ของตัวเองก่อน

ภาพ: หน้า Download ของ Obsidian: ขั้นแรกคือมี vault เล็ก ๆ ของตัวเองก่อน

เปิด Obsidian แล้วสร้าง vault ใหม่

ตั้งชื่อแบบอ่านง่าย เช่น:

```
my-wiki
```

หรือถ้าเป็นงานบริษัท:

```
work-wiki
```

หรือถ้าเป็นโปรเจกต์เฉพาะ:

```
ai-training-wiki
```

อย่าใช้ชื่อกว้างเกินไปแบบ:

```
ทุกอย่าง
สมองที่สอง
โน้ตรวม
```

ชื่อ vault ควรบอกว่าบ้านนี้ใช้ทำอะไร

ถ้าคุณยังไม่แน่ใจ ใช้ `my-wiki` ก่อนก็พอ

สำคัญกว่าชื่อคือการเริ่มใช้จริง

Step 2: สร้าง folder 6 อัน

ใน vault ให้สร้าง folder ตามนี้:

```
00-inbox
10-sources
20-notes
30-concepts
40-projects
90-index
```

ตัวเลขข้างหน้ามีไว้ให้ folder เรียงตามลำดับ

ไม่ใช่เรื่องเทคนิค แค่ช่วยให้ดูง่าย

`00-inbox`

ที่พักของทุกอย่างที่ยังไม่ได้จัด

ใส่ได้ทั้ง:

- meeting note ดิบ
- link ที่เพิ่งเจอ
- idea ที่ยังไม่รู้จะลงไหน
- text ที่ copy มาจากแชท
- note ที่ Claude เพิ่งช่วยร่าง

Inbox คือประตูหลังบ้าน

ของใหม่เข้ามาตรงนี้ก่อน แล้วค่อยจัด

`10-sources`

ที่เก็บ source note

อะไรก็ตามที่เป็นต้นทาง ควรเข้าที่นี่ เช่น:

- บทความ
- PDF
- meeting transcript
- หน้าเว็บคู่แข่ง
- official document
- email สำคัญ

`20-notes`

ที่เก็บ note ทั่วไป

เช่น:

- สรุปประชุมที่จัดแล้ว
- checklist
- observation
- idea ที่ยังไม่เป็น concept
- note ชั่วคราวที่ใช้ทำงาน

`30-concepts`

ที่เก็บแนวคิดที่ใช้ซ้ำ

เช่น:

- Pain point ของลูกค้า SME
- AI training สำหรับคนไม่รู้โค้ด
- Source-first thinking
- Weekly review

นี่คือ folder ที่จะมีค่ามากขึ้นเรื่อย ๆ

`40-projects`

ที่เก็บ project note

เช่น:

- Webinar AI สำหรับผู้บริหาร
- FAQ หน้าเว็บคอร์ส AI
- Knowledge base ทีม sales
- รายงานคู่แข่ง Q2

`90-index`

ที่เก็บหน้า index

เช่น:

- Home
- Index sources
- Index projects
- Index content ideas

ตอนเริ่มมีแค่ Home หน้าเดียวก็พอ

Step 3: สร้าง Home.md

สร้าง note ชื่อ:

> Home

แล้วใส่เนื้อหานี้:

MARKDOWN

```
# Home

นี่คือ LLM Wiki ของฉัน

## ใช้ทำอะไร

- เก็บ source สำคัญ
- สรุป meeting และไอเดีย
- เชื่อมโยงความรู้สำหรับให้ Claude ช่วยต่อ
- ทำ project brief, FAQ, outline และ decision log

## เริ่มจากตรงนี้

- [[00-inbox]]: ของที่ยังไม่ได้จัด
- [[10-sources]]: แหล่งข้อมูลและ reference
- [[20-notes]]: โน้ตทั่วไป
- [[30-concepts]]: แนวคิดที่ใช้ซ้ำ
- [[40-projects]]: งานและโปรเจกต์
- [[90-index]]: หน้า index สำคัญ

## Projects สำคัญ

-

## Concepts สำคัญ

-

## Sources ล่าสุด

-

## Decisions ล่าสุด

-

## กติกาของ Wiki นี้

1. หนึ่ง note หนึ่งเรื่องหลัก
2. ถ้ามี source ให้ใส่ source
3. เรื่องสำคัญต้อง link กลับมาหา project หรือ index
4. inbox ต้องถูกล้างทุกสัปดาห์
5. Claude ช่วยจัดได้ แต่ฉันเป็นคนตรวจสุดท้าย
```

Home คือประตูหน้าบ้าน

เวลาคุณกลับมาเปิด vault หลังจากไม่ได้ใช้หลายวัน ให้เริ่มที่หน้านี้

เวลาคุณให้ Claude ช่วยเข้าใจ vault ก็ส่ง Home ให้มันอ่านก่อน

Step 4: สร้าง Log.md

สร้าง note ชื่อ:

> Log

ใส่ไว้ใน `90-index` หรือไว้หน้า root ก็ได้

เนื้อหาเริ่มต้น:

```markdown
# Log

## YYYY-MM-DD
- สร้าง vault แรก
- สร้าง folder หลัก 6 อัน
- สร้าง Home.md
```

Log ไม่ต้องละเอียด

เขียนแค่พอให้คุณจำได้ว่า wiki เปลี่ยนอะไรไปบ้าง

ถ้าต่อไปให้ Claude ช่วยจัด source ชิ้นหนึ่ง คุณเพิ่ม log แบบนี้:

```markdown
## 2026-05-25
- เพิ่ม source note: [[Meeting ทีม sales เรื่องคำถามลูกค้า AI training]]
- แตก concept note: [[AI training สำหรับคนไม่รู้โค้ด]]
- สร้าง project note: [[Webinar AI สำหรับผู้บริหาร]]
```

Log มีประโยชน์มากตอนกลับมาทำงานต่อ

แทนที่จะถามว่า “ครั้งที่แล้วทำอะไรไว้” คุณเปิด Log แล้วเห็นทันที

Step 5: สร้าง template สำหรับ Source note

สร้าง note ใหม่ชื่อ:

> Template - Source Note

จะเก็บไว้ใน `90-index` หรือสร้าง folder `templates` ก็ได้

ถ้าอยากง่ายที่สุด สร้าง folder เพิ่มชื่อ:

> _templates

แล้วใส่ template ไว้ตรงนั้น

Source template:

```
• MARKDOWN
# ชื่อ source

type: source
status: inbox
date: YYYY-MM-DD
source:

## Summary


## Key facts
-
-
-

## Useful for
-

## Related notes
-

## Questions
-
```

ใช้เมื่อมีข้อมูลจากแหล่งต้นทางหนึ่งชิ้น

เช่นบทความหนึ่งหน้า meeting หนึ่งครั้ง หรือ PDF หนึ่งไฟล์

Step 6: สร้าง template สำหรับ Concept note

สร้าง note ชื่อ:

> Template - Concept Note

ใส่เนื้อหา:

```markdown
# ชื่อ concept

type: concept
status: draft

## ความหมายสั้น ๆ


## ทำไมเรื่องนี้สำคัญ
-

## ตัวอย่าง
-

## ใช้กับงานไหน
-

## Sources
-

## Related concepts
-
```

Concept note ใช้กับเรื่องที่ควรใช้ซ้ำ

ถ้าเป็นแค่ข้อมูลจาก source หนึ่งชิ้น ยังไม่ต้องทำเป็น concept ก็ได้

แต่ถ้าเรื่องนั้นโผล่ซ้ำหลายครั้ง ควรมี concept note

Step 7: สร้าง template สำหรับ Project note

สร้าง note ชื่อ:

> Template - Project Note

ใส่เนื้อหา:

```
• MARKDOWN

# ชื่อโปรเจกต์

type: project
status: active
owner:
deadline:

## เป้าหมาย


## Context สำคัญ
-

## Sources
-

## Concepts ที่เกี่ยวข้อง
-

## Decisions
-

## Todo
- [ ]
- [ ]

## Next review
YYYY-MM-DD
```

Project note คือศูนย์กลางของงานหนึ่งงาน

ถ้าโปรเจกต์มีหลายไฟล์ หลาย source หลาย decision ให้กลับมารวมที่หน้านี้

Step 8: สร้าง template สำหรับ Decision note

สร้าง note ชื่อ:

> Template - Decision Note

ใส่เนื้อหา:

```markdown
# ชื่อ decision

type: decision
date: YYYY-MM-DD
status: decided

## Decision


## Why


## Options considered
-
-

## Evidence / Sources
-

## Impact
-

## Related project
-
```

Decision note ไม่จำเป็นต้องมีทุกวัน

ใช้เฉพาะตอนมีการตัดสินใจที่อาจมีคนลืม เช่น:

- เลือกกลุ่มเป้าหมาย
- เลื่อน deadline
- ตัด scope
- เปลี่ยน message
- เลือกแนวทาง campaign

Step 9: เอาข้อมูลจริง 1 ชิ้นเข้า Inbox

ตอนนี้อย่าเพิ่งเอาทุกอย่างเข้ามา

เลือกข้อมูลจริง 1 ชิ้นพอ

อาจเป็น:

- meeting note ล่าสุด
- บทความที่เกี่ยวกับงาน
- หน้าเว็บคู่แข่ง
- note จาก Claude ที่เคยสรุปไว้
- email ลูกค้าที่สำคัญ

ในตัวอย่างนี้ เราจะใช้ meeting note เดิม:

```
ประชุมทีม sales 25 พ.ค.
ลูกค้าถามบ่อยว่า AI training ต้องรู้โค้ดไหม
หลายคนกลัวทีมใช้ไม่เป็น
ทีม sales อยากได้ FAQ หน้าเว็บ
ควรมี webinar สำหรับผู้บริหารที่ไม่รู้ technical
ต้องส่ง draft FAQ ภายในศุกร์นี้
```

สร้าง note ใน `00-inbox` ชื่อ:

```
Inbox - Meeting ทีม sales 2026-05-25
```

วาง meeting note ดิบลงไป

แค่นี้พอ

อย่าเพิ่งจัดเองทั้งหมด

เราจะให้ Claude ช่วย

Step 10: ให้ Claude ช่วยแปลงเป็น note แรก

เปิด Claude แล้วส่ง meeting note พร้อม prompt นี้:

```
ฉันกำลังตั้งค่า LLM Wiki ใน Obsidian
นี่คือ source แรกใน inbox ของฉัน

ช่วยแปลงเป็น Markdown สำหรับ Obsidian โดยทำ 5 อย่าง:
1. สร้าง Source note จากข้อมูลนี้
2. เสนอ Concept notes ที่ควรสร้าง
3. เสนอ Project notes ที่เกี่ยวข้อง
4. เสนอ Decision note ถ้ามี decision ชัดเจน
5. บอกว่า Home.md ควรเพิ่ม link อะไร

กติกา:
- ใช้เฉพาะข้อมูลที่ฉันให้
- แยก fact / inference / idea ให้ชัด
- ถ้าไม่มีข้อมูลพอ ให้บอกว่าไม่มีข้อมูลพอ
- อย่าแต่งชื่อลูกค้า ตัวเลข หรือรายละเอียดใหม่
- เขียนให้ copy ไปวางใน Obsidian ได้ทันที
```

ผลลัพธ์จาก Claude อาจยาว

คุณไม่จำเป็นต้องใช้ทั้งหมด

เลือกเฉพาะส่วนที่มีประโยชน์และอ่านแล้วถูกต้อง

กติกาคือ:

> Claude เสนอได้ แต่คุณเลือกว่าจะเก็บอะไร

Step 11: สร้าง note จากผลลัพธ์ของ Claude

จาก meeting note นี้ คุณอาจสร้างจริงแค่ 3 note ก่อน:

```
10-sources/Meeting ทีม sales เรื่องคำถามลูกค้า AI training.md
30-concepts/AI training สำหรับคนไม่รู้โค้ด.md
40-projects/FAQ หน้าเว็บคอร์ส AI.md
```

ไม่ต้องสร้างทุก note ที่ Claude เสนอ

ถ้ายังไม่พร้อมทำ webinar ก็ยังไม่ต้องสร้าง project webinar

ถ้า decision ยังไม่ชัด ก็ยังไม่ต้องสร้าง decision note

เริ่มจากน้อย ๆ

Source note ตัวอย่าง

MARKDOWN

```
# Meeting ทีม sales เรื่องคำถามลูกค้า AI training

type: source
status: reviewed
date: 2026-05-25
source: meeting ทีม sales

## Summary
ทีม sales พบว่าลูกค้าถามบ่อยว่า AI training ต้องรู้โค้ดไหม และกังวลว่าทีมจะใช้ AI ไม่เป็น จึงอยากได้ FAQ สำหรับห

## Key facts
- ลูกค้าถามบ่อยว่า AI training ต้องรู้โค้ดไหม
- ลูกค้าหลายคนกลัวทีมใช้ AI ไม่เป็น
- ทีม sales อยากได้ FAQ หน้าเว็บ
- ต้องส่ง draft FAQ ภายในศุกร์นี้

## Useful for
- [[FAQ หน้าเว็บคอร์ส AI]]
- [[AI training สำหรับคนไม่รู้โค้ด]]

## Related notes
- [[Pain point ลูกค้าเรื่องทีมใช้ AI ไม่เป็น]]

## Questions
- ลูกค้าถามด้วยคำจริงว่าอะไรบ้าง
- FAQ ควรมีทั้งหมดกี่ข้อ
```

Concept note ตัวอย่าง

MARKDOWN

```
# AI training สำหรับคนไม่รู้โค้ด

type: concept
status: draft

## ความหมายสั้น ๆ
การสอนใช้ AI กับงานธุรกิจ โดยไม่ตั้งต้นจากการเขียนโปรแกรม แต่ตั้งต้นจากงานจริง เช่น marketing, sales, resea

## ทำไมเรื่องนี้สำคัญ
- ลูกค้าถามบ่อยว่า AI training ต้องรู้โค้ดไหม
- ถ้าสื่อสารไม่ชัด ลูกค้าอาจคิดว่าคอร์สไม่เหมาะกับทีม business

## ตัวอย่าง
- ใช้ Claude สรุป meeting
- ใช้ Claude ทำ FAQ
- ใช้ Claude ช่วยวาง outline webinar

## ใช้กับงานไหน
- [[FAQ หน้าเว็บคอร์ส AI]]

## Sources
- [[Meeting ทีม sales เรื่องคำถามลูกค้า AI training]]

## Related concepts
- [[Pain point ลูกค้าเรื่องทีมใช้ AI ไม่เป็น]]
```

Project note ตัวอย่าง

```markdown
# FAQ หน้าเว็บคอร์ส AI

type: project
status: active
owner: หมิว
deadline: ศุกร์นี้

## เป้าหมาย
สร้าง FAQ หน้าเว็บเพื่อตอบข้อกังวลของลูกค้าว่า AI training ต้องรู้โค้ดหรือไม่ และทีมที่ไม่ technical ใช้ได้ไหม

## Context สำคัญ
- ลูกค้าถามเรื่องต้องรู้โค้ดบ่อย
- ลูกค้าหลายคนกลัวทีมใช้ไม่เป็น
- ทีม sales ต้องการ FAQ สำหรับหน้าเว็บ

## Sources
- [[Meeting ทีม sales เรื่องคำถามลูกค้า AI training]]

## Concepts ที่เกี่ยวข้อง
- [[AI training สำหรับคนไม่รู้โค้ด]]

## Decisions
-

## Todo
- [ ] รวบรวมคำถามจริงจากทีม sales
- [ ] ร่าง FAQ 10 ข้อ
- [ ] ให้ทีม sales ตรวจ

## Next review
YYYY-MM-DD
```

นี่คือ vault แรกที่เริ่มใช้งานจริงแล้ว

ไม่ใช่แค่ folder ว่าง ๆ

Step 12: อัปเดต Home.md

กลับไปที่ `Home.md`

เพิ่ม link สำคัญเข้าไป

```markdown
## Projects สำคัญ

- [[FAQ หน้าเว็บคอร์ส AI]]

## Concepts สำคัญ

- [[AI training สำหรับคนไม่รู้โค้ด]]

## Sources ล่าสุด

- [[Meeting ทีม sales เรื่องคำถามลูกค้า AI training]]
```

Home ต้องไม่กลายเป็นทุกอย่าง

มันเป็นแค่ทางเข้า

ถ้า Home เริ่มยาวเกินไป ให้แยกเป็น index ย่อย เช่น:

> Index Projects
> Index Sources
> Index Concepts

แต่ตอนนี้ยังไม่ต้องทำ

Step 13: อัปเดต Log.md

เปิด `Log.md` แล้วเพิ่ม:

```markdown
## 2026-05-25
- เพิ่ม source note: [[Meeting ทีม sales เรื่องคำถามลูกค้า AI training]]
- สร้าง concept note: [[AI training สำหรับคนไม่รู้โค้ด]]
- สร้าง project note: [[FAQ หน้าเว็บคอร์ส AI]]
- อัปเดต Home.md
```

Log ช่วยให้คุณรู้ว่าเกิดอะไรขึ้นกับ wiki

โดยเฉพาะถ้าคุณให้ Claude ช่วยทำหลายอย่างในอนาคต

Step 14: ล้าง Inbox

เมื่อ source ถูกจัดแล้ว ให้กลับไปที่ `00-inbox`

คุณมี 3 ทางเลือก:

1. ลบ note ดิบ ถ้าข้อมูลถูกย้ายครบแล้ว
2. เก็บไว้เป็น source ดิบใน `10-sources`
3. ย้ายไป folder archive ถ้าคุณอยากเก็บต้นฉบับไว้

สำหรับมือใหม่ แนะนำให้เก็บ source ดิบไว้ก่อน

เพราะถ้า Claude สรุปผิด คุณยังกลับมาตรวจได้

กติกา inbox:

> Inbox มีไว้พัก ไม่ใช่มีไว้กอง

ถ้า inbox มีของค้างเกิน 20 ชิ้น คุณจะเริ่มไม่อยากเปิดมัน

เริ่มจากล้างสัปดาห์ละครั้งก็พอ

สิ่งที่ไม่ต้องทำวันนี้

เพื่อให้คุณไม่หลุดไปแต่งระบบเกินจำเป็น นี่คือสิ่งที่ยังไม่ต้องทำ:

- ยังไม่ต้องติด community plugin
- ยังไม่ต้องทำ Dataview
- ยังไม่ต้องทำ automation
- ยังไม่ต้อง sync ทุกเครื่อง

- ยังไม่ต้องสร้าง property เยอะ
- ยังไม่ต้องทำ folder ซ้อนหลายชั้น
- ยังไม่ต้อง import ทุก note เก่า
- ยังไม่ต้องให้ Claude จัดทั้งชีวิตในวันเดียว

ถ้าคุณอยากติด plugin ในอนาคต ค่อยทำตอนรู้แล้วว่าขาดอะไรจริง ๆ

วันนี้เป้าหมายคือให้ระบบเริ่มเดิน

30-minute setup checklist

ใช้ checklist นี้ทำตามจริงได้เลย

นาทีที่ 0–5: สร้างบ้าน

- ☐ เปิด Obsidian
- ☐ สร้าง vault ใหม่ชื่อ `my-wiki` หรือชื่อที่เหมาะกับงาน
- ☐ เช็กว่าเปิด vault ได้แล้ว

นาทีที่ 5–10: สร้าง folder

- ☐ `00-inbox`
- ☐ `10-sources`
- ☐ `20-notes`
- ☐ `30-concepts`
- ☐ `40-projects`
- ☐ `90-index`
- ☐ `_templates` ถ้าต้องการเก็บ template แยก

นาทีที่ 10–15: สร้างหน้าเริ่มต้น

- ☐ สร้าง `Home.md`
- ☐ ใส่โครง Home จากบทนี้

- ☐ สร้าง `Log.md`

นาทีที่ 15–20: สร้าง template

- ☐ Template - Source Note
- ☐ Template - Concept Note
- ☐ Template - Project Note
- ☐ Template - Decision Note

นาทีที่ 20–25: ใส่ source แรก

- ☐ เลือกข้อมูลจริง 1 ชิ้น
- ☐ วางใน `00-inbox`
- ☐ ส่งให้ Claude พร้อม prompt ในบทนี้

นาทีที่ 25–30: จัด note แรก

- ☐ สร้าง source note อย่างน้อย 1 หน้า
- ☐ สร้าง concept หรือ project note อย่างน้อย 1 หน้า
- ☐ อัปเดต Home
- ☐ อัปเดต Log
- ☐ ล้างหรือย้ายของออกจาก inbox

ถ้าทำครบ คุณมี LLM Wiki รุ่นแรกแล้ว

ไม่ต้องรอให้สมบูรณ์

Prompt ประจำบท: Setup Assistant

ถ้าคุณอยากให้ Claude ช่วยเป็นผู้ช่วยตั้งระบบ ให้ใช้ prompt นี้:

```
ฉันกำลังตั้งค่า Obsidian vault สำหรับทำ LLM Wiki
ฉันไม่รู้โค้ดและอยากได้ระบบที่ง่ายที่สุด

โครงสร้างที่ต้องการ:
- 00-inbox
- 10-sources
- 20-notes
- 30-concepts
- 40-projects
- 90-index

ช่วยทำ 4 อย่าง:
1. ให้เนื้อหา Home.md ที่ copy ไปใช้ได้ทันที
2. ให้ template สำหรับ Source note, Concept note, Project note, Decision note
3. ให้ checklist ใช้งาน 1 สัปดาห์แรก
4. บอกข้อผิดพลาดที่ควรเลี่ยง

กติกา:
- อย่าแนะนำ plugin เว้นแต่จำเป็นจริง
- อย่าใช้ศัพท์ technical
- เขียนให้คนทำงานออฟฟิศทำตามได้
- ให้ผลลัพธ์เป็น Markdown
```

สรุปท้ายบท

บทนี้คุณไม่ได้แค่อ่านเรื่อง Obsidian

คุณสร้างระบบเริ่มต้นแล้ว

ระบบนี้มี:

- vault หนึ่งหลัง
- folder หลัก 6 อัน
- Home สำหรับเริ่มอ่าน
- Log สำหรับบันทึกความเปลี่ยนแปลง
- template สำหรับ note สำคัญ

- source แรกที่ถูกแปลงเป็น wiki note

ถ้าคุณทำตามมาถึงตรงนี้ คุณมีฐานพอสำหรับบทต่อไปแล้ว

บทต่อไปคือหัวใจของการใช้งานจริง:

Workflow: จาก source หนึ่งชิ้นเป็น wiki ที่ใช้ต่อได้

บทที่ 4 คือการตั้งบ้าน

บทที่ 5 คือการใช้บ้านหลังนี้ทุกวัน

Artifact ท้ายบท: Folder เริ่มต้น

```
my-wiki
  00-inbox
  10-sources
  20-notes
  30-concepts
  40-projects
  90-index
  _templates
  Home.md
  Log.md
```

Artifact ท้ายบท: กติกาวันแรก

1. ยังไม่ต้องติด plugin
2. ยังไม่ต้อง import ทุกอย่าง
3. ใช้ข้อมูลจริงแค่ 1 ชิ้นก่อน
4. ให้ Claude ช่วยร่าง แต่ต้องอ่านตรวจ
5. ทำให้ Home และ Log มีชีวิตตั้งแต่วันแรก

อ้างอิงหลักของบทนี้

- Obsidian Help: Download and install Obsidian
- Obsidian Help: Create a vault
- Obsidian Help: Create your first note
- Obsidian Help: Manage vaults
- Obsidian Help: How Obsidian stores data
- Obsidian Help: Properties
- Obsidian Help: Internal links
- Obsidian Help: Community plugins

บทที่ 5

Workflow: จาก source หนึ่งชิ้นเป็น Wiki ที่ใช้ต่อได้

บทที่ 4 เราตั้งบ้านแล้ว

ตอนนี้คุณมี vault มี folder มี Home มี Log และมี template พื้นฐาน

บทนี้คือหัวใจของการใช้จริง:

> ทำยังไงให้ข้อมูลหนึ่งชิ้น กลายเป็นความรู้ที่ใช้ต่อได้

ข้อมูลหนึ่งชิ้นอาจเป็นอะไรก็ได้:

- meeting note
- บทความ
- PDF
- email ลูกค้า
- หน้าเว็บคู่แข่ง
- transcript จาก call
- สรุปที่ Claude เคยทำไว้
- ไอเดียที่คุณจดตอนเดินทาง

ถ้าคุณแค่โยนทั้งหมดเข้า Obsidian มันจะกลายเป็นกองของที่หาไม่เจอ

แต่ถ้าคุณใช้ workflow ที่ชัด ข้อมูลหนึ่งชิ้นจะกลายเป็น:

- source note ที่มีที่มา
- concept note ที่ใช้ซ้ำได้
- project note ที่ทำงานต่อได้
- decision note ที่กันลืม

- link ใน Home หรือ index ที่หาเจอ

นี่คือจุดที่ LLM Wiki เริ่มมีประโยชน์จริง

Workflow ทั้งบทใน 7 ขั้น

Workflow หลักคือเปลี่ยน source หนึ่งชิ้นให้กลายเป็น Wiki ที่ใช้ต่อได้

ภาพ: Workflow หลักคือเปลี่ยน source หนึ่งชิ้นให้กลายเป็น Wiki ที่ใช้ต่อได้

จำแค่ 7 คำนี้พอ:

> `Capture → Clean → Ask Claude → Split → Link → Save → Review`

แปลเป็นภาษาคนทำงาน:

1. **Capture**: เก็บข้อมูลเข้า inbox
2. **Clean**: ทำให้ข้อมูลอ่านได้

3. **Ask Claude**: ให้ Claude ช่วยสรุปและจัดรูป
4. **Split**: แยกเป็น note ที่ถูกชนิด
5. **Link**: เชื่อมกับ note เดิม
6. **Save**: ย้ายเข้าที่และอัปเดต Home/Log
7. **Review**: อ่านตรวจ ไม่เชื่อ AI แบบอัตโนมัติ

หนึ่ง source ใช้เวลาประมาณ 10–20 นาที

ถ้าเป็น source สำคัญมาก อาจนานกว่านั้น

แต่ถ้า source ธรรมดาแล้วใช้เวลาเป็นชั่วโมง แปลว่าคุณกำลังทำละเอียดเกินไป

ตัวอย่าง source ของบทนี้

เราจะใช้ตัวอย่างเดิมจากบทก่อน เพื่อให้เห็นภาพต่อเนื่อง

นี่คือ source ดิบ:

> ประชุมทีม sales 25 พ.ค.
> ลูกค้าถามบ่อยว่า AI training ต้องรู้โค้ดไหม
> หลายคนกลัวทีมใช้ไม่เป็น
> ทีม sales อยากได้ FAQ หน้าเว็บ
> ควรมี webinar สำหรับผู้บริหารที่ไม่รู้ technical
> ต้องส่ง draft FAQ ภายในศุกร์นี้

Source นี้สั้นมาก แต่มีข้อมูลหลายแบบปนกัน:

- fact จากทีม sales
- pain point ของลูกค้า
- idea เรื่อง webinar
- project เรื่อง FAQ
- deadline

ถ้าจดไว้เฉย ๆ มันเป็นแค่บันทึกประชุม

ถ้าแปลงเป็น LLM Wiki มันกลายเป็น knowledge asset

Step 1: Capture: เก็บเข้าที่เดียวก่อน

เวลาเจอข้อมูลใหม่ อย่าเพิ่งคิดว่าจะจัดลง folder ไหน

ให้โยนเข้า `00-inbox` ก่อน

สร้าง note ชื่อ:

> `Inbox - Meeting ทีม sales 2026-05-25`

ใส่ source ดิบลงไปแบบนี้:

```
• MARKDOWN

# Inbox - Meeting ทีม sales 2026-05-25

type: inbox
status: raw
date: 2026-05-25
source: meeting ทีม sales

## Raw note

ประชุมทีม sales 25 พ.ค.
ลูกค้าถามบ่อยว่า AI training ต้องรู้โค้ดไหม
หลายคนกลัวทีมใช้ไม่เป็น
ทีม sales อยากได้ FAQ หน้าเว็บ
ควรมี webinar สำหรับผู้บริหารที่ไม่รู้ technical
ต้องส่ง draft FAQ ภายในศุกร์นี้
```

ทำไมต้องเก็บแบบ raw ก่อน?

เพราะตอน capture เป้าหมายคือ “อย่าให้ข้อมูลหาย”

ยังไม่ต้องจัดสวย

ยังไม่ต้องคิดชื่อ concept

ยังไม่ต้องทำ project note

จับให้ได้ก่อน

จัดทีหลัง

Step 2: Clean: ทำให้ source อ่านได้

ก่อนส่งให้ Claude อ่าน ลองเช็ก 5 อย่าง:

1. มีวันที่ไหม
2. มีที่มาไหม
3. มีบริบทสั้น ๆ ไหมว่า source นี้คืออะไร
4. มีคำย่อหรือชื่อเฉพาะที่ Claude อาจไม่เข้าใจไหม
5. มีข้อมูลที่ไม่ควรส่งให้ AI ไหม

ตัวอย่าง source ดิบเมื่อกี้ ยังขาด context นิดหน่อย

ปรับเป็นแบบนี้ดีกว่า:

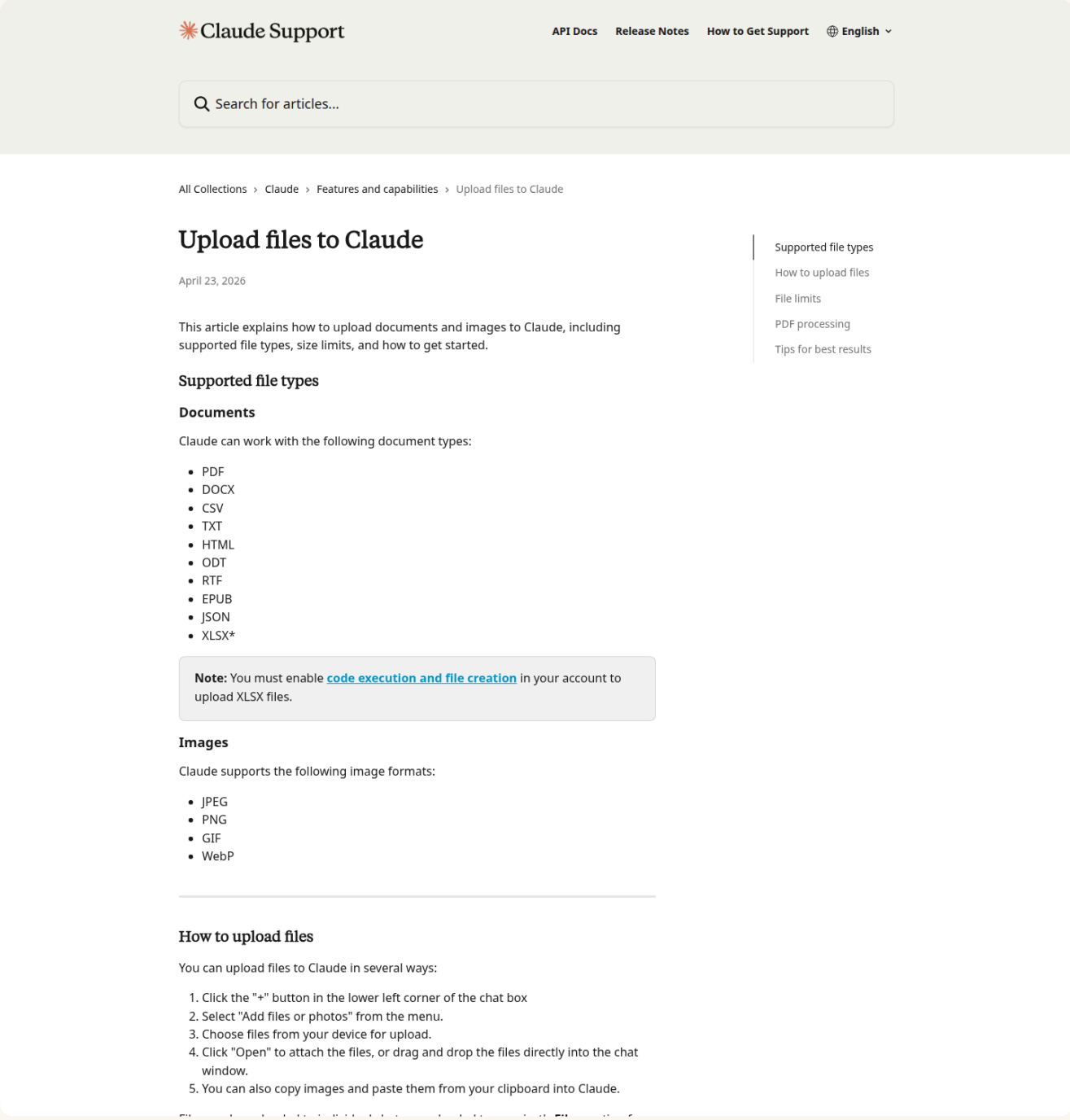
```markdown
# Inbox - Meeting ทีม sales 2026-05-25

type: inbox
status: raw
date: 2026-05-25
source: internal sales meeting
context: บริษัทกำลังขายคอร์ส AI training สำหรับเจ้าของธุรกิจและทีม business ที่ไม่จำเป็นต้องรู้โค้ด

## Raw note

ประชุมทีม sales 25 พ.ค.
ลูกค้าถามบ่อยว่า AI training ต้องรู้โค้ดไหม
หลายคนกลัวทีมใช้ไม่เป็น
ทีม sales อยากได้ FAQ หน้าเว็บ
ควรมี webinar สำหรับผู้บริหารที่ไม่รู้ technical
ต้องส่ง draft FAQ ภายในศุกร์นี้
```

เพิ่มแค่บรรทัด `context` Claude ก็เข้าใจดีขึ้นมาก

Clean ไม่ได้แปลว่าต้องเขียนใหม่ทั้งหมด

แค่ทำให้ source ไม่กำกวม

Step 3: Ask Claude: ให้โจทย์ชัด ไม่ใช่แค่ “สรุปให้หน่อย”



หน้า Claude Support เรื่องการ upload files: Claude อ่านไฟล์ได้ แต่เราต้องเลือก context ให้พอดี

ภาพ: หน้า Claude Support เรื่องการ upload files: Claude อ่านไฟล์ได้ แต่เราต้องเลือก context ให้พอดี

Prompt ที่อ่อนเกินไป:

```
ช่วยสรุป meeting นี้ให้หน่อย
```

Claude อาจสรุปได้ แต่ยังไม่พร้อมเข้า Wiki

Prompt ที่ควรใช้:

```
ฉันกำลังทำ LLM Wiki ใน Obsidian
นี่คือ source จาก meeting ทีม sales

ช่วยแปลง source นี้เป็นชุด note สำหรับ wiki โดยทำ 7 อย่าง:
1. สรุป source เป็น Source note
2. แยก fact / inference / idea
3. เสนอ Concept notes ที่ควรสร้าง
4. เสนอ Project notes ที่เกี่ยวข้อง
5. เสนอ Decision note ถ้ามีการตัดสินใจชัดเจน
6. เสนอ internal links ที่ควรใช้
7. บอกว่า Home.md และ Log.md ควรอัปเดตอะไร

กติกา:
- ใช้เฉพาะข้อมูลที่ให้มา
- อย่าแต่งชื่อคน ชื่อลูกค้า ตัวเลข หรือรายละเอียดใหม่
- ถ้าข้อมูลไม่พอ ให้เขียนว่า "ข้อมูลไม่พอ"
- เขียนเป็น Markdown ที่ copy ไปใช้ใน Obsidian ได้
- ใช้ภาษาคนทำงานทั่วไป ไม่ต้อง technical
```

สังเกตว่า prompt นี้ไม่ได้ขอแค่ summary

แต่ขอให้ Claude ช่วยจัดระบบ

นี่คือความต่างใหญ่

Step 4: อ่านผลลัพธ์ของ Claude แบบ editor

หลัง Claude ตอบ อย่า copy ทั้งหมดทันที

อ่านเหมือนคุณตรวจงานผู้ช่วย

ใช้ checklist นี้:

```
[ ] มีข้อมูลที่ Claude แต่งเพิ่มไหม
[ ] fact กับ inference แยกกันไหม
[ ] ชื่อ note อ่านรู้เรื่องไหม
[ ] มี note ที่ไม่จำเป็นหรือเยอะเกินไปไหม
[ ] link ที่เสนอมีประโยชน์จริงไหม
[ ] deadline หรือ decision ถูกต้องไหม
[ ] มีอะไรควรถามทีม/ลูกค้าเพิ่มไหม
```

Claude อาจเสนอ note เยอะเกินไป เช่น:

```
- ลูกค้ากังวลเรื่อง AI
- AI training
- ทีม sales
- FAQ
- Webinar
- ผู้บริหาร
- ไม่รู้โค้ด
```

ไม่ต้องสร้างทั้งหมด

ถามตัวเองว่า:

> note นี้อนาคตจะกลับมาใช้ไหม?

ถ้าไม่แน่ใจ อย่าเพิ่งสร้าง

LLM Wiki ที่ดีต้องกล้าตัด

Step 5: Split: แยกเป็น note ที่ถูกชนิด

จาก source ตัวอย่าง เราควรสร้างแค่ 3–4 note ก่อน

1. Source note

> `10-sources/Meeting ทีม sales เรื่องคำถามลูกค้า AI training.md`

เพราะ meeting นี้คือ source ต้นทาง

2. Concept note

> `30-concepts/AI training สำหรับคนไม่รู้โค้ด.md`

เพราะเรื่องนี้น่าจะใช้ซ้ำใน FAQ, webinar, landing page และ sales script

3. Project note

> `40-projects/FAQ หน้าเว็บคอร์ส AI.md`

เพราะทีม sales ขอ output ชัดเจน

4. Project หรือ idea note อีกอัน ถ้าจำเป็น

> `40-projects/Webinar AI สำหรับผู้บริหาร.md`

แต่ถ้ายังไม่เริ่มทำจริง ให้ใส่ไว้ใน section `Idea` ก่อน ยังไม่ต้องสร้าง project note

Decision note ต้องมีไหม?

source บอกว่า “ควรมี webinar” แต่ยังไม่ชัดว่าใครตัดสินใจแล้ว

ดังนั้นยังไม่ควรสร้าง decision note แบบมั่นใจ

ให้เขียนใน source note ว่า:

```
• MARKDOWN

## Idea
- อาจทำ webinar สำหรับผู้บริหารที่ไม่ technical
```

ไม่ใช่:

```markdown
## Decision
ตัดสินใจทำ webinar แล้ว
```

นี่คือการแยก fact/inference/idea ในชีวิตจริง

Step 6: เขียน Source note ให้แน่น

Source note ที่ดีไม่ต้องยาว แต่ต้องตอบว่า source นี้มีอะไรและมาจากไหน

ตัวอย่าง final version:

```markdown
# Meeting ทีม sales เรื่องคำถามลูกค้า AI training

type: source
status: reviewed
date: 2026-05-25
source: internal sales meeting

## Context
บริษัทกำลังขายคอร์ส AI training สำหรับเจ้าของธุรกิจและทีม business ที่ไม่จำเป็นต้องรู้โค้ด

## Summary
ทีม sales พบว่าลูกค้าถามบ่อยว่า AI training ต้องรู้โค้ดไหม และกังวลว่าทีมจะใช้ AI ไม่เป็น ทีมจึงอยากได้ FAQ สำหร

## Fact
- ลูกค้าถามบ่อยว่า AI training ต้องรู้โค้ดไหม
- ลูกค้าหลายคนกลัวทีมใช้ AI ไม่เป็น
- ทีม sales อยากได้ FAQ หน้าเว็บ
- ต้องส่ง draft FAQ ภายในศุกร์นี้

## Inference
- ลูกค้าอาจเข้าใจว่า AI training = programming
- เนื้อหาการขายควรย้ำว่าเหมาะกับทีม business ที่ไม่ technical

## Idea
- อาจทำ webinar สำหรับผู้บริหารที่ไม่ technical

## Useful for
- [[FAQ หน้าเว็บคอร์ส AI]]
- [[AI training สำหรับคนไม่รู้โค้ด]]

## Questions
- คำถามจริงจากลูกค้ามี wording ว่าอะไร
- FAQ ควรตอบกี่ข้อ
- Webinar เป็น decision แล้วหรือยัง เป็นแค่ idea
```

จุดสำคัญคือ note นี้ไม่พยายามทำทุกอย่าง

มันเก็บ source ให้ชัด แล้วชี้ไป note อื่น

Step 7: เขียน Concept note ให้ใช้ซ้ำได้

Concept note ควรสั้นและคม

อย่าเอาทุกอย่างจาก source มาใส่ซ้ำ

ตัวอย่าง:

```
• MARKDOWN

# AI training สำหรับคนไม่รู้โค้ด

type: concept
status: draft

## ความหมายสั้น ๆ
AI training สำหรับคนไม่รู้โค้ด คือการสอนใช้ AI กับงานธุรกิจ โดยเริ่มจากงานจริง เช่น meeting, email, sales,

## ทำไมเรื่องนี้สำคัญ
- ลูกค้าบางส่วนกังวลว่าการเรียน AI ต้องมีพื้นฐาน code
- ถ้าสื่อสารไม่ชัด ลูกค้าอาจคิดว่าคอร์สไม่เหมาะกับทีม business
- แนวคิดนี้ควรปรากฏใน FAQ, landing page และ webinar

## ตัวอย่าง use case
- ใช้ Claude สรุป meeting
- ใช้ Claude ช่วยร่าง FAQ
- ใช้ Claude วิเคราะห์คำถามลูกค้า
- ใช้ Claude ทำ outline webinar

## Sources
- [[Meeting ทีม sales เรื่องคำถามลูกค้า AI training]]

## Related
- [[FAQ หน้าเว็บคอร์ส AI]]
```

Concept note มีหน้าที่เป็น “ความเข้าใจกลาง”

ถ้ามี source ใหม่ในอนาคต เช่น feedback จากลูกค้า หรือ survey ทีม sales คุณมาอัปเดต concept note นี้ได้

ความรู้จึงค่อย ๆ สะสม

Step 8: เขียน Project note ให้ทำงานต่อได้

Project note ต้องช่วยให้ลงมือทำต่อ ไม่ใช่แค่สรุปสวย

ตัวอย่าง:

MARKDOWN

```
# FAQ หน้าเว็บคอร์ส AI

type: project
status: active
owner: หมิว
deadline: ศุกร์นี้

## เป้าหมาย
สร้าง FAQ หน้าเว็บเพื่อตอบข้อกังวลของลูกค้าว่า AI training ต้องรู้โค้ดหรือไม่ และทีม business ใช้ได้จริงไหม

## Context สำคัญ
- ลูกค้าถามบ่อยว่า AI training ต้องรู้โค้ดไหม
- ลูกค้าหลายคนกลัวทีมใช้ AI ไม่เป็น
- ทีม sales ต้องการ FAQ สำหรับหน้าเว็บ

## Sources
- [[Meeting ทีม sales เรื่องคำถามลูกค้า AI training]]

## Concepts
- [[AI training สำหรับคนไม่รู้โค้ด]]

## Draft FAQ ideas
- ต้องรู้โค้ดก่อนไหม
- ถ้าทีมไม่เคยใช้ AI จะเรียนทันไหม
- ใช้กับงาน sales/marketing/admin ได้ไหม
- หลังอบรมจะได้ output อะไร

## Todo
- [ ] ขอคำถามจริงจากทีม sales
- [ ] ร่าง FAQ 10 ข้อ
- [ ] ให้ทีม sales ตรวจ wording
- [ ] ส่งให้คนดูแลเว็บอัปเดตหน้า landing page

## Next review
YYYY-MM-DD
```

Project note ที่ดีทำให้คุณกลับมาทำต่อได้ทันที

ไม่ต้องอ่าน source ใหม่ทั้งหมด

Step 9: Link: เชื่อมเท่าที่จำเป็น

ก่อนสร้างหรือเชื่อม note ใหม่ ให้ search คำหลักใน Obsidian สั้น ๆ ก่อน

ถ้ามี note เดิมอยู่แล้ว ให้ link ไปหา note เดิม แทนที่จะสร้าง note ซ้ำอีกหน้า

หลังสร้าง note แล้ว เช็ก link หลัก ๆ

ใน source note ควร link ไป:

```
[[AI training สำหรับคนไม่รู้โค้ด]]
[[FAQ หน้าเว็บคอร์ส AI]]
```

ใน concept note ควร link กลับไป source:

```
[[Meeting ทีม sales เรื่องคำถามลูกค้า AI training]]
```

ใน project note ควร link ไป source และ concept:

```
[[Meeting ทีม sales เรื่องคำถามลูกค้า AI training]]
[[AI training สำหรับคนไม่รู้โค้ด]]
```

อย่าทำ link เยอะจนอ่านยาก

กติกาง่าย ๆ:

> Link เฉพาะเรื่องที่ถ้าคลิกไปแล้วช่วยให้เข้าใจงานต่อ

คำทั่วไปไม่ต้อง link

เช่นคำว่า “ลูกค้า”, “AI”, “เว็บ”, “ทีม” ไม่จำเป็นต้อง link ทุกครั้ง

Step 10: Save: ย้ายเข้าที่

ตอนนี้ย้าย note ไป folder ที่ถูกต้อง

```
10-sources/Meeting ทีม sales เรื่องคำถามลูกค้า AI training.md
30-concepts/AI training สำหรับคนไม่รู้โค้ด.md
40-projects/FAQ หน้าเว็บคอร์ส AI.md
```

แล้วจัดการ note ใน inbox

มี 3 ทางเลือก:

1. ลบ inbox note ถ้าข้อมูลย้ายครบแล้ว
2. เก็บไว้เป็น raw source
3. ย้ายไป `10-sources/raw` ถ้าคุณอยากเก็บต้นฉบับแยก

สำหรับมือใหม่ แนะนำให้เก็บ raw source ไว้ก่อน

เพราะมันช่วยตรวจว่า Claude สรุปผิดไหม

Step 11: อัปเดต Home

เปิด `Home.md` แล้วเพิ่ม link สำคัญ

```markdown
## Projects สำคัญ
- [[FAQ หน้าเว็บคอร์ส AI]]

## Concepts สำคัญ
- [[AI training สำหรับคนไม่รู้โค้ด]]

## Sources ล่าสุด
- [[Meeting ทีม sales เรื่องคำถามลูกค้า AI training]]
```

ถ้า Home มี link มากขึ้นเรื่อย ๆ อย่าตกใจ

แต่ถ้าเริ่มยาวเกินอ่าน ให้แยก index ย่อยในบทหลัง ๆ

ตอนนี้แค่ให้ Home เป็นทางเข้าได้ก็พอ

Step 12: อัปเดต Log

เปิด `Log.md` แล้วเพิ่ม:

```markdown
## 2026-05-25
- Processed source: [[Meeting ทีม sales เรื่องคำถามลูกค้า AI training]]
- Created concept: [[AI training สำหรับคนไม่รู้โค้ด]]
- Created project: [[FAQ หน้าเว็บคอร์ส AI]]
- Updated Home.md
- Open question: webinar เป็น decision แล้วหรือยัง
```

Log ที่ดีไม่ต้องยาว

แค่บอกว่าเกิดอะไรขึ้น และมีคำถามอะไรค้างอยู่

Step 13: Review: ตรวจ 5 นาทีสุดท้าย

ก่อนจบ workflow ให้ตรวจ 5 นาที

ใช้ checklist นี้:

> [ ] Source note มีที่มาชัดไหม
> [ ] Fact / inference / idea แยกกันไหม
> [ ] Concept note ซ้ำกับ source เกินไปไหม
> [ ] Project note มี next action ชัดไหม
> [ ] Link สำคัญครบไหม
> [ ] Home อัปเดตไหม
> [ ] Log อัปเดตไหม
> [ ] มีคำถามค้างที่ต้องถามคนจริงไหม

ถ้าครบ แปลว่า source นี้ถูกแปลงเป็น wiki แล้ว

ไม่ใช่แค่ถูกเก็บไว้

Prompt หลักของบทนี้

นี่คือ prompt ที่ใช้ซ้ำได้กับ source แทบทุกแบบ

```
ฉันกำลังทำ LLM Wiki ใน Obsidian
ช่วยแปลง source นี้เป็น wiki notes ที่ใช้ต่อได้จริง

ข้อมูล context:
- งาน/โปรเจกต์ที่เกี่ยวข้อง:
- กลุ่มคนที่เกี่ยวข้อง:
- เป้าหมายของการเก็บ source นี้:

สิ่งที่ต้องการ:
1. Source note พร้อม Summary, Fact, Inference, Idea, Questions
2. Concept notes ที่ควรสร้างหรืออัปเดต
3. Project note ที่เกี่ยวข้อง ถ้ามี
4. Decision note ถ้ามี decision ชัดเจนเท่านั้น
5. Internal links ที่ควรเชื่อม
6. Home.md ควรเพิ่ม link อะไร
7. Log.md ควรบันทึกอะไร

กติกา:
- ใช้เฉพาะข้อมูลที่ฉันให้
- อย่าแต่งข้อมูลให้ดูสมบูรณ์
- ถ้าข้อมูลไม่พอ ให้เขียนว่า "ข้อมูลไม่พอ"
- แยก fact / inference / idea ให้ชัด
- ถ้าเป็นแค่ idea อย่าเขียนเป็น decision
- ใช้ชื่อ note ที่คนทำงานทั่วไปอ่านแล้วเข้าใจ
- เขียนเป็น Markdown ที่ copy ไปใช้ใน Obsidian ได้
```

คุณสามารถเก็บ prompt นี้ไว้ใน Obsidian เป็น note ชื่อ:

```
Prompt - Source to Wiki
```

แล้วใช้ซ้ำทุกครั้ง

Workflow สำหรับ source แบบต่าง ๆ

Source แต่ละแบบใช้หลักเดียวกัน แต่มีจุดที่ต้องระวังต่างกัน

1. Meeting note

ให้จับ 5 อย่าง:

- ใครประชุม
- คุยเรื่องอะไร
- fact คืออะไร
- decision คืออะไร
- todo คืออะไร

ระวัง:

- อย่าให้ Claude เดาว่าใครรับผิดชอบ ถ้าใน note ไม่ได้เขียน
- อย่าเปลี่ยน idea ให้กลายเป็น decision

2. บทความ

ให้จับ 5 อย่าง:

- บทความนี้พูดเรื่องอะไร
- key claims คืออะไร
- source น่าเชื่อแค่ไหน
- ใช้กับงานเราได้ตรงไหน
- ควรแตก concept อะไร

ระวัง:

- อย่า copy บทความยาว ๆ มาแทนความเข้าใจ
- อย่าเชื่อทุก claim ถ้าไม่มี source รองรับ

3. หน้าเว็บคู่แข่ง

ให้จับ 5 อย่าง:

- เขาขายอะไร
- message หลักคืออะไร
- กลุ่มเป้าหมายคือใคร
- เขาตอบ objection อะไร
- เราเรียนรู้อะไรได้

ระวัง:

- อย่าลอก wording
- แยก observation กับ strategy ของเรา

4. Email ลูกค้า

ให้จับ 5 อย่าง:

- ลูกค้าถามอะไร
- pain point คืออะไร
- ต้องตอบกลับอะไร
- เกี่ยวกับ project ไหน
- มีข้อมูล sensitive ไหม

ระวัง:

- ลบข้อมูลส่วนตัวที่ไม่จำเป็นก่อนส่งให้ AI
- อย่าให้ Claude ตอบในนามบริษัทโดยไม่ตรวจ

5. Chat เก่ากับ Claude

ให้จับ 5 อย่าง:

- คุยเรื่องอะไร
- ได้ข้อสรุปอะไร
- มี prompt หรือ output ที่ควรเก็บไหม
- อะไรเป็นแค่ draft
- อะไรควรกลายเป็น concept note

ระวัง:

- Chat เก่าอาจมีคำตอบผิดปนอยู่
- ให้เก็บเฉพาะสิ่งที่ตรวจแล้วหรือมีประโยชน์จริง

กติกา “หนึ่ง source ไม่ต้องสร้างสิบ note เสมอ”

ใน LLM Wiki บางแนวทาง source หนึ่งชิ้นอาจแตกเป็นหลายหน้า

แต่สำหรับคนทำงานทั่วไป อย่าเริ่มแบบนั้น

ให้ใช้กติกานี้:

```
source ธรรมดา → 1 source note
source ที่มี pattern ใช้ซ้ำ → +1 concept note
source ที่เกี่ยวกับงานจริง → +1 project note
source ที่มี decision ชัด → +1 decision note
```

แปลว่า source หนึ่งชิ้นอาจได้แค่ note เดียวก็ได้

ไม่ผิด

อย่าสร้าง note เพื่อให้ระบบดูฉลาด

สร้าง note เมื่อมันช่วยให้คุณทำงานต่อได้

วิธีรู้ว่า source นี้ควรแตก concept หรือไม่

ถามตัวเอง 4 ข้อ:

1. เรื่องนี้จะโผล่อีกไหม
2. จะใช้กับมากกว่าหนึ่ง project ไหม
3. ถ้า Claude รู้ concept นี้ จะช่วยงานเราได้ดีขึ้นไหม
4. มี source มากกว่าหนึ่งชิ้นรองรับหรือมีโอกาสจะมีเพิ่มไหม

ถ้าตอบ “ใช่” อย่างน้อย 2 ข้อ ค่อยสร้าง concept note

ถ้าไม่ใช่ เก็บไว้ใน source note ก่อน

วิธีรู้ว่า source นี้ควรสร้าง project note หรือไม่

ถามตัวเอง:

1. มี output ที่ต้องทำต่อไหม
2. มี deadline ไหม
3. มีคนรับผิดชอบไหม
4. มี todo มากกว่าหนึ่งข้อไหม
5. ต้องกลับมาดูซ้ำไหม

ถ้าใช่ ให้สร้าง project note

ถ้าเป็นแค่ไอเดียลอย ๆ ยังไม่ต้อง

ใส่ไว้ใน section `Idea` ของ source note ก่อน

วิธีรู้ว่า source นี้ควรสร้าง decision note หรือไม่

Decision note ต้องใช้เมื่อมีการตัดสินใจจริง

คำที่บอกว่าอาจเป็น decision:

- ตกลงว่า...
- เลือก...
- ไม่ทำ...
- เลื่อนไป...
- ตัดออก...
- ให้ใช้แนวทาง...

คำที่ยังไม่ใช่ decision:

- น่าจะ...
- ควรลอง...
- อาจทำ...
- มีไอเดียว่า...
- ทีมเสนอว่า...

ถ้ายังไม่ชัด ให้เขียนว่า `Idea` หรือ `Open question`

อย่ารีบเขียนเป็น decision

ความผิดพลาดที่ควรเลี่ยง

1. เก็บเยอะ แต่ไม่ process

ถ้า inbox มีแต่ของดิบ Wiki จะไม่ช่วยอะไร

ต้องมีเวลาสั้น ๆ สำหรับ process

2. ให้ Claude สร้าง note เยอะเกินไป

Claude ขยันมาก

แต่คุณไม่จำเป็นต้องเก็บทุกอย่างที่มันเสนอ

3. ไม่เก็บ source

สรุปที่ไม่มี source จะตรวจยาก

โดยเฉพาะเมื่อใช้กับงานจริง

4. ไม่อัปเดต Home

ถ้า note สำคัญไม่อยู่ใน Home หรือ index วันหลังจะหาไม่เจอ

5. ไม่แยก idea กับ decision

นี่ทำให้งานสับสนที่สุด

idea คือสิ่งที่อาจทำ

decision คือสิ่งที่ตกลงแล้ว

6. ทำระบบใหญ่กว่างาน

ถ้าจัด note นานกว่าทำงานจริง ระบบกำลังเริ่มผิดทาง

ใช้เวลาเท่าไรต่อ source หนึ่งชิ้น

สำหรับ source ธรรมดา:

```
Capture: 1–2 นาที
Clean: 2–3 นาที
Ask Claude: 2 นาที
Review output: 3–5 นาที
Save + link + log: 5 นาที
```

รวมประมาณ 15 นาที

สำหรับ source สำคัญ อาจใช้ 30–45 นาที

แต่ไม่ควรทุกชิ้น

ให้เลือกทำละเอียดเฉพาะ source ที่มีผลกับงานจริง

เลือกโหมดตามเวลาที่มี

Workflow เดียวกัน ไม่ได้แปลว่าต้องใช้เวลานานเท่ากันทุกครั้ง

โหมด 15 นาที

ใช้กับ source ธรรมดา เช่น meeting note สั้น ๆ email หนึ่งฉบับ หรือบทความที่ไม่ได้สำคัญมาก

ทำแค่ 5 อย่าง:

```
1. เก็บ source พร้อม date/source/context
2. ให้ Claude สรุป fact / inference / idea
3. สร้างหรืออัปเดต note ที่จำเป็นที่สุด 1-2 หน้า
4. ใส่ link กลับไป source
5. อัปเดต Log สั้น ๆ
```

โหมดนี้เน้น “พอใช้ต่อได้” ไม่ใช่สมบูรณ์แบบ

โหมด 45 นาที

ใช้กับ source สำคัญ เช่น research สำคัญ feedback ลูกค้าหลายราย strategy meeting หรือเอกสารที่กระทบงานใหญ่

ทำเพิ่มจากโหมด 15 นาที:

```
1. ตรวจ source ดิบละเอียดขึ้น
2. แยก concept ที่จะใช้ซ้ำจริง
3. อัปเดต project note ให้มี next action
4. เช็กว่า decision มีจริงไหม
5. ให้ Claude ช่วยหาช่องว่างหรือคำถามที่ควรถามต่อ
6. อัปเดต Home หรือ index ให้หาเจอ
```

โหมดนี้ใช้เมื่อ source มีผลกับงานจริง

อย่าใช้โหมด 45 นาทีกับทุกอย่าง

ถ้าทุก source ต้องละเอียดหมด คุณจะเลิกทำภายในสองสัปดาห์

สรุปท้ายบท

LLM Wiki ไม่ได้เกิดจากการเก็บทุกอย่าง

มันเกิดจากการ process source ทีละชิ้นให้กลายเป็นความรู้ที่ใช้ต่อได้

Workflow หลักคือ:

```
Capture → Clean → Ask Claude → Split → Link → Save → Review
```

ถ้าทำซ้ำได้สัปดาห์ละไม่กี่ครั้ง Wiki จะเริ่มโตแบบมีคุณภาพ

และเมื่อ Wiki โตขึ้น Claude จะช่วยได้ดีขึ้น เพราะมันไม่ได้อ่านกองข้อมูลดิบ แต่มันอ่านความรู้ที่คุณค่อย ๆ จัดไว้แล้ว

บทต่อไป เราจะดูเรื่องการดูแล Wiki ไม่ให้รก

เพราะระบบความรู้ไม่ได้พังตอนเริ่ม

มันพังหลังจากใช้ไป 3 สัปดาห์แล้วไม่เคย cleanup

Artifact ท้ายบท: Workflow 7 ขั้น

```
1. Capture      เก็บข้อมูลเข้า inbox
2. Clean        ทำให้ source อ่านรู้เรื่อง
3. Ask Claude   ให้ Claude ช่วยสรุป/จัดรูป
4. Split        แยกเป็น note ที่ถูกชนิด
5. Link         เชื่อมกับ note เดิม
6. Save         ย้ายเข้าที่ อัปเดต Home/Log
7. Review       อ่านตรวจก่อนถือว่าเสร็จ
```

Artifact ท้ายบท: Source processing checklist

```
[ ] Source มีวันที่และที่มา
[ ] เพิ่ม context สั้น ๆ แล้ว
[ ] ส่งให้ Claude ด้วย prompt ที่ชัด
[ ] ตรวจว่า Claude ไม่แต่งข้อมูลเพิ่ม
[ ] แยก fact / inference / idea แล้ว
[ ] สร้าง note เท่าที่จำเป็น
[ ] เชื่อม link สำคัญ
[ ] อัปเดต Home
[ ] อัปเดต Log
[ ] มี open questions ถ้าข้อมูลยังขาด
```

อ้างอิงหลักของบทนี้

- Andrej Karpathy: LLM Wiki gist: ingest, query, lint, index/log pattern
- `AgriciDaniel/claude-obsidian`: LLM Wiki Pattern note
- `AgriciDaniel/claude-obsidian`: Source-First Synthesis note
- Obsidian Help: Internal links
- Obsidian Help: Properties
- Claude Help Center: Upload files to Claude
- Claude Help Center: Projects and project knowledge

บทที่ 6

ใช้ Claude ดูแล Wiki ไม่ให้รก

บทที่ 5 เราเรียน workflow หลักแล้ว

ข้อมูลหนึ่งชิ้นเข้ามา → แปลงเป็น source note → แตกเป็น concept/project เท่าที่จำเป็น → link → save → review

ถ้าทำแบบนี้ไปเรื่อย ๆ Wiki จะโต

ปัญหาคือ Wiki ที่โตขึ้น ไม่ได้แปลว่า Wiki ดีขึ้นเสมอ

ถ้าไม่ดูแล มันจะเริ่มเป็นแบบนี้:

- inbox เต็มไปด้วย note ดิบ
- note ชื่อคล้ายกันหลายหน้า
- project เก่าค้างอยู่ แต่ไม่มีใครปิด
- Home มี link เยอะจนอ่านไม่ออก
- concept บางหน้าดีมาก แต่หาไม่เจอ
- decision เก่าไม่ตรงกับสิ่งที่ทีมทำจริงแล้ว
- Claude อ่านแล้วงง เพราะ note ขัดกันเอง

นี่คือจุดที่หลายคนเลิกใช้ระบบความรู้

ไม่ใช่เพราะ Obsidian ไม่ดี

ไม่ใช่เพราะ Claude ไม่เก่ง

แต่เพราะไม่มีเวลาจัดบ้าน

บทนี้จะสอนวิธีดูแล Wiki แบบคนทำงานทั่วไป

สัปดาห์ละครั้ง 30 นาทีก็พอ

หลักคิดของบทนี้

จำประโยคนี้ไว้:

> Claude ช่วยตรวจบ้านได้ แต่คุณต้องเป็นคนตัดสินใจว่าจะย้ายอะไร ทิ้งอะไร และเชื่ออะไร

Claude เหมาะกับงานแบบนี้:

- อ่าน note หลายหน้าแล้วหาจุดซ้ำ
- ชี้ว่า note ไหนไม่มี link
- ชี้ว่า project ไหนดูเหมือนค้าง
- สรุปว่า week นี้ Wiki เปลี่ยนอะไร
- เสนอว่า Home ควรจัดใหม่ยังไง
- ถามคำถามที่เราลืมถาม

แต่ Claude ไม่ควรทำแทนทั้งหมด

เพราะมันไม่รู้บริบทจริงเท่าคุณ

มันไม่รู้ว่า project ไหนเลิกแล้วจริง ๆ

มันไม่รู้ว่า decision ไหนเปลี่ยนเพราะคุยนอก meeting

ดังนั้นบทนี้ใช้ Claude เป็น “ผู้ช่วยตรวจบ้าน” ไม่ใช่เจ้าของบ้าน

Wiki รกมีหน้าตายังไง

Wiki รกไม่ได้แปลว่ามี note เยอะ

Wiki รกคือ Wiki ที่กลับมาใช้ไม่ได้

อาการหลักมี 6 แบบ

1. Inbox ไม่เคยว่าง

`00-inbox` มีของดิบเต็มไปหมด

บางอันเป็น meeting note

บางอันเป็น link

บางอันเป็นไอเดียครึ่งบรรทัด

บางอันจำไม่ได้แล้วว่าเก็บมาทำไม

ถ้า inbox ไม่เคยถูก process มันจะกลายเป็นถังขยะสุภาพ

2. Note ซ้ำกันหลายชื่อ

เช่น:

```
AI training สำหรับ non-tech.md
AI training ไม่ต้องเขียนโค้ด.md
คอร์ส AI สำหรับคนทั่วไป.md
อบรม AI สำหรับ business team.md
```

ทุกหน้าอาจพูดเรื่องใกล้กัน

แต่ไม่มีหน้าไหนเป็นตัวหลัก

เวลาถาม Claude มันอาจดึงหลายหน้ามาปนกัน

3. Note ไม่มี link

Note บางหน้าเหมือนเกาะลอยน้ำ

ไม่มีใคร link มาหา

ไม่มี link ออกไปหาใคร

ถ้าเป็น concept หรือ project สำคัญ แล้วไม่มี link แปลว่ามีปัญหา

4. Home ยาวเกินไป

ตอนแรก Home ช่วยให้เริ่มต้นง่าย

แต่หลังใช้ไปสักพัก Home อาจกลายเป็นหน้าที่รวมทุกอย่าง

สุดท้าย Home ไม่ใช่ประตูหน้าบ้านแล้ว

มันกลายเป็นห้องเก็บของอีกห้อง

5. Project ค้าง

Project note บางหน้า status ยังเป็น `active`

แต่จริง ๆ จบไปแล้ว

หรือหยุดไปแล้ว

หรือเปลี่ยนชื่อไปแล้ว

ถ้าไม่ปิด project เก่า ระบบจะสร้างความรู้สึกว่างานค้างเต็มไปหมด

6. Decision เก่าไม่ถูกอัปเดต

Decision note มีประโยชน์มาก

แต่ถ้า decision เปลี่ยนแล้วไม่บันทึก จะอันตราย

เช่นเคยตัดสินใจว่า:

> จะทำ webinar สำหรับผู้บริหารเดือนมิถุนายน

แต่ต่อมาทีมเปลี่ยนเป็นทำ e-book แทน

ถ้า decision note เก่ายังอยู่โดยไม่บอกว่าเปลี่ยนแล้ว Claude อาจสรุปผิดว่า webinar ยังเป็นแผนหลัก

Weekly Review: ดูแล Wiki สัปดาห์ละครั้ง

Weekly Review คือวงรอบสั้น ๆ ที่ทำให้ Wiki ไม่รกและกลับมาใช้ต่อได้

ภาพ: Weekly Review คือวงรอบสั้น ๆ ที่ทำให้ Wiki ไม่รกและกลับมาใช้ต่อได้

วิธีง่ายที่สุดคือทำ Weekly Review

เลือกเวลาตายตัว เช่น:

- ศุกร์ 16:30
- จันทร์ 09:30
- หลังประชุมทีมประจำสัปดาห์

ใช้เวลา 30 นาที

เป้าหมายคือทำให้ Wiki กลับมาใช้ต่อได้ในสัปดาห์หน้า

Weekly Review มี 6 ขั้น:

```
1. ล้าง inbox
2. เช็ก project ที่ active
3. หา note ซ้ำ
4. หา note ที่ไม่มี link
5. อัปเดต Home/Index
6. เขียน Log สรุปสัปดาห์
```

Step 1: ล้าง inbox

เปิด `00-inbox`

แบ่งของใน inbox เป็น 4 กอง

```
Process now    = ควรแปลงเป็น wiki note ตอนนี้
Keep raw       = เก็บไว้ก่อน ยังไม่ต้องทำ
Delete         = ไม่ใช้แล้ว ลบทิ้ง
Ask later      = ต้องถามคนอื่นก่อน
```

ตัวอย่าง inbox ของหมิว:

```
Inbox - Meeting sales 2026-05-25.md
Inbox - ลูกค้าถามเรื่องราคา.txt
Inbox - ไอเดีย webinar ผู้บริหาร.md
Inbox - link บทความ AI adoption.md
Inbox - meeting note ที่จดไม่ครบ.md
```

หมิวอาจจัดแบบนี้:

```
Process now:
- ลูกค้าถามเรื่องราคา

Keep raw:
- link บทความ AI adoption

Ask later:
- meeting note ที่จดไม่ครบ

Delete:
- ไอเดีย webinar ผู้บริหาร ถ้าซ้ำกับ source เดิมแล้ว
```

กติกาคือ:

> Inbox ไม่จำเป็นต้องว่าง 100% แต่ต้องไม่เป็นที่ซ่อนของงานที่ควรทำ

ถ้า inbox มี 3–5 ชิ้น แต่รู้สถานะทุกชิ้น ยังรับได้

ถ้ามี 50 ชิ้นและไม่รู้ว่าอะไรคืออะไร แปลว่าต้องหยุดเก็บเพิ่มแล้ว process ก่อน

Prompt: ให้ Claude ช่วยล้าง inbox

```
ฉันกำลังทำ Weekly Review ของ LLM Wiki ใน Obsidian
นี่คือรายการ note ใน 00-inbox พร้อมเนื้อหาสั้น ๆ

ช่วยจัดกลุ่มแต่ละ note เป็น 4 ประเภท:
1. Process now: ควรแปลงเป็น wiki note ตอนนี้
2. Keep raw: เก็บไว้ก่อน ยังไม่ต้องทำ
3. Delete: ไม่จำเป็นแล้วหรือซ้ำชัดเจน
4. Ask later: ต้องถามข้อมูลเพิ่มก่อน

กติกา:
- อย่าลบอะไรแทนฉัน
- ถ้าไม่แน่ใจ ให้ใส่ Ask later
- อธิบายเหตุผลสั้น ๆ ต่อ note
- ถ้า note ไหนควรกลายเป็น Source/Concept/Project/Decision ให้บอกด้วย
```

Claude ช่วยลดแรงตัดสินใจ

แต่คุณเป็นคนกดลบ ย้าย หรือ process เอง

Step 2: เช็ก project ที่ active

เปิด folder `40-projects`

มองหา project ที่มี status เป็น `active` หรือ `waiting`

ถามตัวเอง 5 ข้อ:

```
[ ] project นี้ยังทำอยู่จริงไหม
[ ] deadline ยังถูกไหม
[ ] next action ชัดไหม
[ ] มี source ล่าสุดครบไหม
[ ] ควรปิด ย้าย หรือรอไหม
```

ตัวอย่าง project ของหมิว:

```markdown
# FAQ หน้าเว็บคอร์ส AI

type: project
status: active
owner: หมิว
deadline: ศุกร์นี้
```

ตอน Weekly Review หมิวควรเช็กว่า:

- FAQ draft ส่งแล้วหรือยัง
- ทีม sales ตรวจแล้วหรือยัง
- deadline ผ่านไปหรือยัง
- ต้องเปลี่ยน status เป็น `done` หรือ `waiting` ไหม

Status ที่แนะนำแบบง่าย:

```
draft     = เพิ่งเริ่ม ยังไม่ชัด
active    = กำลังทำ
waiting   = รอคนอื่น
done      = เสร็จแล้ว
paused    = หยุดไว้ก่อน
archived  = เก็บเป็นประวัติ ไม่ใช้ทำงานต่อ
```

ไม่ต้องมี status เยอะกว่านี้

ถ้าเยอะเกิน คนจะไม่ใช้

Prompt: ให้ Claude ช่วยตรวจ project

```
ฉันกำลัง review project notes ใน Obsidian
ต่อไปนี้คือ project notes ที่ status ยังเป็น active หรือ waiting

ช่วยตรวจให้หน่อยว่า:
1. project ไหนยังมี next action ชัด
2. project ไหนควรเปลี่ยน status เป็น waiting/done/paused/archived
3. project ไหนขาด source หรือ context
4. project ไหนควรถามคนจริงเพิ่ม
5. มี project ไหนซ้ำหรือควรรวมกันไหม

กติกา:
- อย่าเดาว่างานเสร็จแล้วถ้าใน note ไม่บอก
- ถ้าไม่แน่ใจ ให้เขียนว่า "ต้องตรวจเอง"
- เสนอเป็น checklist สั้น ๆ ต่อ project
```

Claude เหมาะกับการอ่านแล้วสะกิด

เช่น:

```
FAQ หน้าเว็บคอร์ส AI
- status อาจยัง active แต่ deadline "ศุกร์นี้" ควรเปลี่ยนเป็นวันที่จริง
- next action ชัด: ขอคำถามจริงจากทีม sales
- ขาด source: feedback หลังส่ง draft
- ต้องตรวจเอง: ทีม sales อนุมัติ FAQ หรือยัง
```

Step 3: หา note ซ้ำ

Note ซ้ำเป็นเรื่องปกติ

เวลาทำงานจริง เรามักตั้งชื่อเรื่องเดียวกันคนละแบบในคนละวัน

สิ่งที่ต้องทำคือเลือก “หน้าหลัก”

ตัวอย่าง:

```
AI training สำหรับคนไม่รู้โค้ด.md
AI สำหรับ business team.md
อบรม AI แบบไม่ technical.md
```

ให้เลือกหนึ่งหน้าเป็นหน้าหลัก เช่น:

```
AI training สำหรับคนไม่รู้โค้ด.md
```

อีกสองหน้าอาจทำได้ 3 แบบ:

1. รวมเนื้อหาเข้า note หลัก แล้วลบหน้าเก่า
2. เปลี่ยนหน้าเก่าเป็น note สั้น ๆ ที่ link ไปหน้าหลัก
3. เก็บไว้ ถ้ามันมีมุมเฉพาะจริง ๆ

สำหรับมือใหม่ แนะนำแบบที่ 2

เพราะปลอดภัยกว่า

• MARKDOWN

```
# AI สำหรับ business team

หัวข้อนี้รวมอยู่ที่ [[AI training สำหรับคนไม่รู้โค้ด]]

หน้านี้เก็บไว้เพื่อเป็นทางเข้าอีกชื่อหนึ่ง
```

แบบนี้ถ้าวันหลัง search ด้วยคำว่า business team ก็ยังเจอ

แต่ความรู้หลักอยู่หน้าเดียว

Prompt: ให้ Claude ช่วยหา note ซ้ำ

```
นี่คือรายชื่อ note ใน Obsidian ของฉัน
ช่วยหากลุ่ม note ที่อาจซ้ำหรือใกล้กันเกินไป

สิ่งที่ต้องการ:
1. จัดกลุ่ม note ที่ชื่อคล้ายกันหรือพูดเรื่องใกล้กัน
2. เสนอว่า note ไหนควรเป็นหน้าหลัก
3. เสนอ note ไหนควรรวม/เปลี่ยนเป็นทางเข้า/ปล่อยไว้
4. ระบุว่าข้อเสนอไหนต้องเปิดเนื้อหาดูเพิ่มก่อน

กติกา:
- อย่าฟันธงจากชื่ออย่างเดียวถ้าไม่พอ
- ใช้คำว่า "อาจซ้ำ" ไม่ใช่ "ซ้ำแน่นอน"
- เป้าหมายคือทำให้ Wiki ใช้ง่ายขึ้น ไม่ใช่ลบให้เหลือน้อยที่สุด
```

Step 4: หา note ที่ไม่มี link

ใน Obsidian คุณจะเห็น backlink ได้

ถ้า note ไม่มีใคร link มา และไม่มี link ออกไป ให้ถามว่า:

> note นี้ควรอยู่ตรงไหนในบ้าน?

Note ไม่มี link มี 3 แบบ

1. Source note ธรรมดา

ถ้าเป็น source ที่เก็บไว้เป็นหลักฐาน อาจไม่เป็นไร

แต่ถ้ามันสำคัญ ควร link ไป project หรือ concept อย่างน้อยหนึ่งหน้า

2. Concept สำคัญแต่ลอยอยู่

อันนี้ควรแก้

เช่น `AI training สำหรับคนไม่รู้โค้ด` ควร link กับ:

```
[[FAQ หน้าเว็บคอร์ส AI]]
[[Meeting ทีม sales เรื่องคำถามลูกค้า AI training]]
```

3. Note ที่ไม่มีประโยชน์แล้ว

ถ้าอ่านแล้วไม่รู้ว่าเก็บไว้ทำไม อาจ archive หรือลบ

ถ้าไม่แน่ใจ ให้ใส่ status:

```
status: review-later
```

อย่าเสียเวลาตัดสินใจทุกอย่างในครั้งเดียว

Prompt: ให้ Claude ช่วยดู link ที่ขาด

```
ฉันมี note ที่ดูเหมือนยังไม่มี link ชัดเจน
ช่วยดูว่า note นี้ควรเชื่อมกับ note ไหนใน Wiki

ข้อมูลที่ให้:
1. เนื้อหา note ที่สงสัย
2. รายชื่อ note สำคัญใน Home/Index
3. project ที่กำลัง active

สิ่งที่ต้องการ:
- เสนอ internal links ที่ควรเพิ่ม
- บอกเหตุผลสั้น ๆ ต่อ link
- ถ้าไม่ควร link ก็ให้บอกว่าไม่จำเป็น
- เสนอว่า note นี้ควรเป็น Source/Concept/Project/Decision หรือ archive

กติกา:
- อย่าเสนอ link เพียงเพราะมีคำเหมือนกัน
- link ต้องช่วยให้กลับมาใช้ทำงานได้จริง
```

Step 5: อัปเดต Home/Index

Home คือหน้าเริ่มต้น

ถ้า Home ไม่อัปเดต Wiki จะเริ่มหาอะไรไม่เจอ

แต่ Home ก็ไม่ควรแบกทุกอย่าง

กติกาง่าย ๆ:

> Home = สิ่งที่ต้องเห็นบ่อย
> Index = แผนที่ของหมวดใดหมวดหนึ่ง
> Search = ใช้หาเรื่องเฉพาะ

ถ้า Home เริ่มยาว ให้แยก index ย่อย

ตัวอย่าง:

```
90-index/Sales and Customer Questions.md
90-index/AI Training Content.md
90-index/Active Projects.md
```

Home อาจเหลือแค่:

• MARKDOWN

```markdown
# Home

## Start here
- [[Active Projects]]
- [[Sales and Customer Questions]]
- [[AI Training Content]]

## Projects สำคัญตอนนี้
- [[FAQ หน้าเว็บคอร์ส AI]]

## Review
- [[Log]]
```

Home ที่ดีไม่ใช่ Home ที่ครบที่สุด

Home ที่ดีคือ Home ที่พาคุณกลับไปทำงานได้เร็วที่สุด

Prompt: ให้ Claude ช่วยจัด Home

> นี่คือ Home.md ปัจจุบันของ LLM Wiki และรายชื่อ note สำคัญบางส่วน
> ช่วยเสนอวิธีจัด Home ให้สั้นและใช้เริ่มงานได้ดีขึ้น
>
> สิ่งที่ต้องการ:
> 1. ส่วนไหนควรอยู่ใน Home ต่อ
> 2. ส่วนไหนควรย้ายไป index แยก
> 3. ควรสร้าง index note อะไรบ้าง
> 4. ตัวอย่าง Home.md เวอร์ชันใหม่แบบสั้น
>
> กติกา:
> - Home ต้องอ่านจบใน 1 นาที
> - อย่าใส่ทุก note ลง Home
> - ใช้ภาษาคนทำงานทั่วไป
> - ถ้าไม่แน่ใจ ให้เสนอเป็น option ไม่ฟันธง

Step 6: เขียน Weekly Log

หลัง review เสร็จ ให้เขียน Log สั้น ๆ

ตัวอย่าง:

```markdown
## 2026-05-29: Weekly Review

### Processed
- Processed inbox note: ลูกค้าถามเรื่องราคา
- Updated project: [[FAQ หน้าเว็บคอร์ส AI]]
- Created concept: [[ราคาและความคุ้มค่าของ AI training]]

### Cleaned
- Merged notes about AI training for non-technical users into [[AI training สำหรับคนไม่รู้
- Moved old webinar idea to review-later

### Open questions
- ทีม sales อนุมัติ FAQ แล้วหรือยัง
- Webinar ยังเป็นแผนอยู่ไหม หรือเปลี่ยนเป็น e-book

### Next week
- ขอ feedback จากทีม sales
- อัปเดต FAQ draft หลัง review
```

Log ไม่ได้มีไว้โชว์

มันมีไว้ให้คุณและ Claude เห็นว่า Wiki เปลี่ยนอะไรไปแล้ว

เวลาอีก 2 เดือนกลับมาดู จะรู้ว่าเรื่องนี้เคยเกิดขึ้นตอนไหน

Weekly Review Prompt แบบครบชุด

ใช้เมื่อคุณมีเวลาสัปดาห์ละครั้ง

ให้เลือกส่งเฉพาะ note ที่เกี่ยวข้อง เช่น:

- Home.md
- Log.md ล่าสุด
- รายชื่อ note ใน inbox
- project notes ที่ active
- concept notes ที่สงสัยว่าซ้ำ

Prompt:

```
ฉันกำลังทำ Weekly Review ของ LLM Wiki ใน Obsidian
เป้าหมายคือทำให้ Wiki ไม่รก หาเจอ และใช้ต่อกับงานจริงได้

ข้อมูลที่ฉันให้:
- Home.md ปัจจุบัน
- Log.md ล่าสุด
- รายชื่อ note ใน 00-inbox
- project notes ที่ active/waiting
- รายชื่อ concept notes บางส่วน

ช่วย review 6 เรื่อง:
1. Inbox: note ไหนควร process / keep raw / delete / ask later
2. Projects: project ไหนควร active / waiting / done / paused / archived
3. Duplicates: note ไหนอาจซ้ำหรือควรรวม
4. Links: note ไหนดูเหมือนขาด link สำคัญ
5. Home/Index: Home ควรปรับอะไร และควรสร้าง index note อะไร
6. Questions: มีคำถามอะไรที่ต้องถามคนจริงก่อนตัดสินใจ

กติกา:
- อย่าเดาข้อมูลที่ไม่มีใน note
- ถ้าไม่แน่ใจ ให้เขียนว่า "ต้องตรวจเอง"
- อย่าเสนอให้ลบอะไรถ้าเหตุผลไม่ชัด
- แยก fact / inference / suggestion ให้ชัด
- เขียนเป็น checklist ที่ฉันทำตามได้ใน 30 นาที
```

เก็บ prompt นี้ไว้เป็น note ชื่อ:

```
Prompt - Weekly Wiki Review
```

ตัวอย่าง Weekly Review ของหมิว

วันศุกร์ หมิวเปิด Wiki แล้วเจอว่า:

```
00-inbox
- ลูกค้าถามเรื่องราคา
- ไอเดีย webinar ผู้บริหาร
- link บทความ AI adoption

40-projects
- FAQ หน้าเว็บคอร์ส AI.md status: active
- Webinar AI สำหรับผู้บริหาร.md status: draft

30-concepts
- AI training สำหรับคนไม่รู้โค้ด.md
- AI สำหรับ business team.md
```

หมิวส่งรายการนี้พร้อม Home และ project notes ให้ Claude

Claude อาจเสนอว่า:

```
Inbox
- ลูกค้าถามเรื่องราคา: Process now เพราะเกี่ยวกับ FAQ
- ไอเดีย webinar ผู้บริหาร: Ask later เพราะยังไม่รู้ว่าเป็น decision หรือ idea
- link บทความ AI adoption: Keep raw ถ้ายังไม่เกี่ยวกับงานสัปดาห์นี้

Projects
- FAQ หน้าเว็บคอร์ส AI: ยัง active แต่ควรเพิ่ม next action และวันที่จริง
- Webinar AI สำหรับผู้บริหาร: ยังเป็น draft/idea ไม่ควร active ถ้ายังไม่มี
owner/deadline

Duplicates
- AI training สำหรับคนไม่รู้โค้ด กับ AI สำหรับ business team อาจซ้ำบางส่วน
- แนะนำให้ใช้ AI training สำหรับคนไม่รู้โค้ด เป็นหน้าหลัก

Home
- เพิ่ม link ไป FAQ หน้าเว็บคอร์ส AI
- ย้าย source ล่าสุดไป Log แทน ไม่ต้องแสดงใน Home ทั้งหมด

Questions
- Webinar เป็นแผนจริงหรือยัง
- FAQ draft ส่งให้ทีม sales แล้วหรือยัง
```

หมิวเลือกทำ 4 อย่าง:

1. Process note เรื่องราคาเป็น source note
2. อัปเดต FAQ project ให้มี next action
3. เปลี่ยน Webinar จาก `draft` เป็น `idea` ยังไม่ active
4. รวม concept ซ้ำให้มีหน้าหลักหน้าเดียว

แค่นี้ Weekly Review ก็สำเร็จแล้ว

ไม่ต้องทำให้ Wiki สะอาดเหมือนเริ่มใหม่

ทำให้สัปดาห์หน้ากลับมาใช้ได้ก็พอ

กติกา 30 นาที

Weekly Review ไม่ควรกลายเป็นงานใหญ่อีกงาน

ใช้ timer 30 นาที

แบ่งเวลาแบบนี้:

```
5 นาที   ดู inbox
7 นาที   ดู project active/waiting
5 นาที   ดู note ซ้ำหรือ note ลอย
5 นาที   ปรับ Home/Index
5 นาที   เขียน Log
3 นาที   จดคำถามที่ต้องถามคนจริง
```

ถ้าทำไม่เสร็จ ให้จบด้วย note ชื่อ:

```
Review Later.md
```

ใส่รายการค้างไว้ เช่น:

```markdown
# Review Later

## ต้องดูต่อ
- concept note เรื่อง pricing อาจซ้ำกับ value proposition
- source จากบทความ AI adoption ยังไม่ได้ process
- webinar idea ต้องถามทีม sales ก่อน
```

นี่ดีกว่าฝืนทำต่อจนเหนื่อย

ระบบที่ดีต้องอยู่กับชีวิตจริงได้

อย่าเอาทั้ง vault ให้ Claude ทุกครั้ง

มือใหม่มักคิดว่า:

> ถ้าให้ Claude อ่านทั้งหมด มันจะช่วยได้ดีที่สุด

ไม่เสมอไป

Claude ทำงานดีเมื่อ context ชัด

ให้คิดแบบนี้:

```
ถามเรื่อง project → ส่ง project note + source ที่เกี่ยวข้อง + concept สำคัญ
ถามเรื่อง cleanup → ส่ง Home + inbox list + project active + note ที่สงสัย
ถามเรื่อง concept → ส่ง concept note + sources ที่รองรับ
```

อย่าส่งเพราะกลัวพลาด

ส่งเพราะมันเกี่ยวกับคำถาม

นี่คือวิธีใช้ Claude แบบคุม context

เมื่อ Claude เสนอให้ลบหรือรวม note

Claude อาจเสนอว่า note นี้ซ้ำ ควรรวม หรือลบ

ให้ใช้กติกานี้ก่อนทำจริง:

```
[ ] เปิด note อ่านเองแล้วหรือยัง
[ ] มี source สำคัญอยู่ใน note นี้ไหม
[ ] มี link จาก note อื่นมาหรือเปล่า
[ ] ถ้าลบ จะเสีย context อะไรไหม
[ ] ถ้าไม่แน่ใจ archive แทนลบได้ไหม
```

ถ้าไม่แน่ใจ ให้ archive ก่อน

สร้าง folder ง่าย ๆ:

```
99-archive
```

หรือใช้ status:

```
status: archived
```

Archive คือ “ไม่ใช้ทำงานตอนนี้ แต่ยังเก็บไว้ได้”

ลบเมื่อมั่นใจจริง ๆ

ใช้ Properties และ Tag แค่พอช่วยหา

ไม่ต้องใช้ properties เยอะ

แต่มี 4 ช่องที่ช่วยมาก:

```MARKDOWN
type: project
status: active
date: 2026-05-25
review: 2026-05-29
```

Properties มีไว้ให้เห็นสถานะเร็ว

Tag ใช้ได้ แต่อย่าให้เยอะ

แนะนำ tag กว้าง ๆ เช่น:

> #ลูกค้า
> #sales
> #content
> #idea
> #ต้องถามต่อ

อย่าสร้าง tag ละเอียดแบบนี้:

> #ลูกค้าถามว่าAItrainingต้องรู้โค้ดไหม

เรื่องแบบนั้นควรเป็นชื่อ note หรือ link ไม่ใช่ tag

Tag มีไว้รวมกลุ่มกว้าง ๆ

Link มีไว้เชื่อมความหมายจริง

Properties มีไว้บอกสถานะ

สัญญาณว่า Wiki เริ่มสุขภาพดี

หลังทำ Weekly Review ไป 2–3 รอบ คุณจะเริ่มเห็นสิ่งนี้:

- inbox มีของค้างน้อยลง
- project active มีจำนวนที่รับมือได้

- Home สั้นลง แต่ใช้ง่ายขึ้น
- concept สำคัญมี source รองรับ
- decision เก่าถูกปิดหรืออัปเดต
- Claude ตอบงานได้ตรงขึ้น เพราะ note ไม่ขัดกันเอง
- คุณหาเรื่องเดิมเจอเร็วขึ้น

นี่คือเป้าหมาย

ไม่ใช่ graph สวย

ไม่ใช่ note เยอะ

ไม่ใช่ระบบซับซ้อน

แต่คือกลับมาใช้ความรู้ของตัวเองได้เร็วขึ้น

ความผิดพลาดที่ควรเลี่ยง

1. Review ทุกหน้า

ไม่จำเป็น

ดูเฉพาะ inbox, active projects, Home และ note ที่มีปัญหา

2. Clean จนไม่เหลือเวลาทำงานจริง

ถ้าใช้เวลาจัด Wiki มากกว่าทำงาน Wiki เริ่มผิดจุด

3. ให้ Claude ตัดสินใจแทน

Claude เสนอได้

แต่คนตัดสินใจคือคุณ

4. ลบเร็วเกินไป

ถ้าไม่แน่ใจ archive ก่อน

5. ทำ Home ให้ครบทุกอย่าง

Home มีไว้เริ่ม ไม่ใช่รวมทั้งจักรวาล

6. ไม่เขียน Log

ถ้าไม่เขียน Log คุณจะไม่รู้ว่าทำอะไรไปแล้ว

โดยเฉพาะเวลาใช้ Claude ช่วยจัดหลายรอบ

สรุปท้ายบท

Wiki ที่ดีไม่ได้ดีเพราะตั้งระบบครั้งแรกสวย

มันดีเพราะถูกดูแลนิดหน่อยอย่างสม่ำเสมอ

ใช้หลักนี้พอ:

```
Weekly Review 30 นาที
- ล้าง inbox
- เช็ก project
- หา note ซ้ำ
- หา note ไม่มี link
- ปรับ Home/Index
- เขียน Log
```

Claude ช่วยอ่าน ช่วยเสนอ ช่วยเตือนจุดที่คุณมองข้าม

แต่คุณยังเป็นเจ้าของบ้าน

ถ้าบทที่ 5 คือวิธีทำให้ Wiki โต

บทนี้คือวิธีทำให้ Wiki โตแล้วไม่รก

บทต่อไป เราจะพูดเรื่องสุดท้ายก่อนเอาไปใช้จริง:

5 กติกากันพลาด

เพราะระบบนี้จะมีประโยชน์มากขึ้นเมื่อใช้กับงานจริง

และยิ่งใช้กับงานจริง ยิ่งต้องมีขอบเขตที่ชัด

Artifact ท้ายบท: Weekly Review Checklist

```
[ ] เปิด 00-inbox
[ ] จัด note เป็น Process now / Keep raw / Delete / Ask later
[ ] เปิด project ที่ status active/waiting
[ ] อัปเดต status, deadline, next action
[ ] หา note ที่ชื่อหรือเนื้อหาใกล้กันเกินไป
[ ] เลือก note หลักถ้ามีเรื่องซ้ำ
[ ] หา note สำคัญที่ไม่มี link
[ ] เพิ่ม link เฉพาะที่ช่วยทำงานต่อ
[ ] ทำ Home ให้สั้นและเริ่มงานได้เร็ว
[ ] สร้าง index ย่อยถ้า Home ยาวเกินไป
[ ] เขียน Weekly Log
[ ] จดคำถามที่ต้องถามคนจริง
```

Artifact ท้ายบท: Weekly Review Prompt

```
ฉันกำลังทำ Weekly Review ของ LLM Wiki ใน Obsidian
เป้าหมายคือทำให้ Wiki ไม่รก หาเจอ และใช้ต่อกับงานจริงได้

ข้อมูลที่ฉันให้:
- Home.md ปัจจุบัน
- Log.md ล่าสุด
- รายชื่อ note ใน 00-inbox
- project notes ที่ active/waiting
- รายชื่อ concept notes บางส่วน

ช่วย review 6 เรื่อง:
1. Inbox: note ไหนควร process / keep raw / delete / ask later
2. Projects: project ไหนควร active / waiting / done / paused / archived
3. Duplicates: note ไหนอาจซ้ำหรือควรรวม
4. Links: note ไหนดูเหมือนขาด link สำคัญ
5. Home/Index: Home ควรปรับอะไร และควรสร้าง index note อะไร
6. Questions: มีคำถามอะไรที่ต้องถามคนจริงก่อนตัดสินใจ

กติกา:
- อย่าเดาข้อมูลที่ไม่มีใน note
- ถ้าไม่แน่ใจ ให้เขียนว่า "ต้องตรวจเอง"
- อย่าเสนอให้ลบอะไรถ้าเหตุผลไม่ชัด
- แยก fact / inference / suggestion ให้ชัด
- เขียนเป็น checklist ที่ฉันทำตามได้ใน 30 นาที
```

อ้างอิงหลักของบทนี้

- Obsidian Help: Internal links และ backlinks
- Obsidian Help: Properties
- Obsidian Help: Tags
- Obsidian Help: Search
- Claude Help Center: Upload files to Claude
- Claude Help Center: Projects and project knowledge
- Andrej Karpathy: LLM Wiki gist: lint, index, log, update pattern

- `AgriciDaniel/claude-obsidian`: LLM Wiki Pattern และ Source-First Synthesis

บทที่ 7

ก่อนเอาไปใช้จริง: 5 กติกากันพลาด

ถึงตรงนี้ คุณมีภาพรวมครบแล้ว

Obsidian คือบ้านของความรู้

Claude คือผู้ช่วยอ่าน คิด และจัดบ้าน

LLM Wiki คือวิธีเขียนความรู้ให้ทั้งคนและ AI อ่านรู้เรื่อง

บทที่ 4 พาคุณตั้งบ้าน

บทที่ 5 พาคุณเอา source หนึ่งชิ้นเข้าระบบ

บทที่ 6 พาคุณดูแลบ้านไม่ให้รก

บทนี้คือบทสั้น ๆ ก่อนเอาไปใช้กับงานจริง

ไม่ใช่บทกฎหมาย

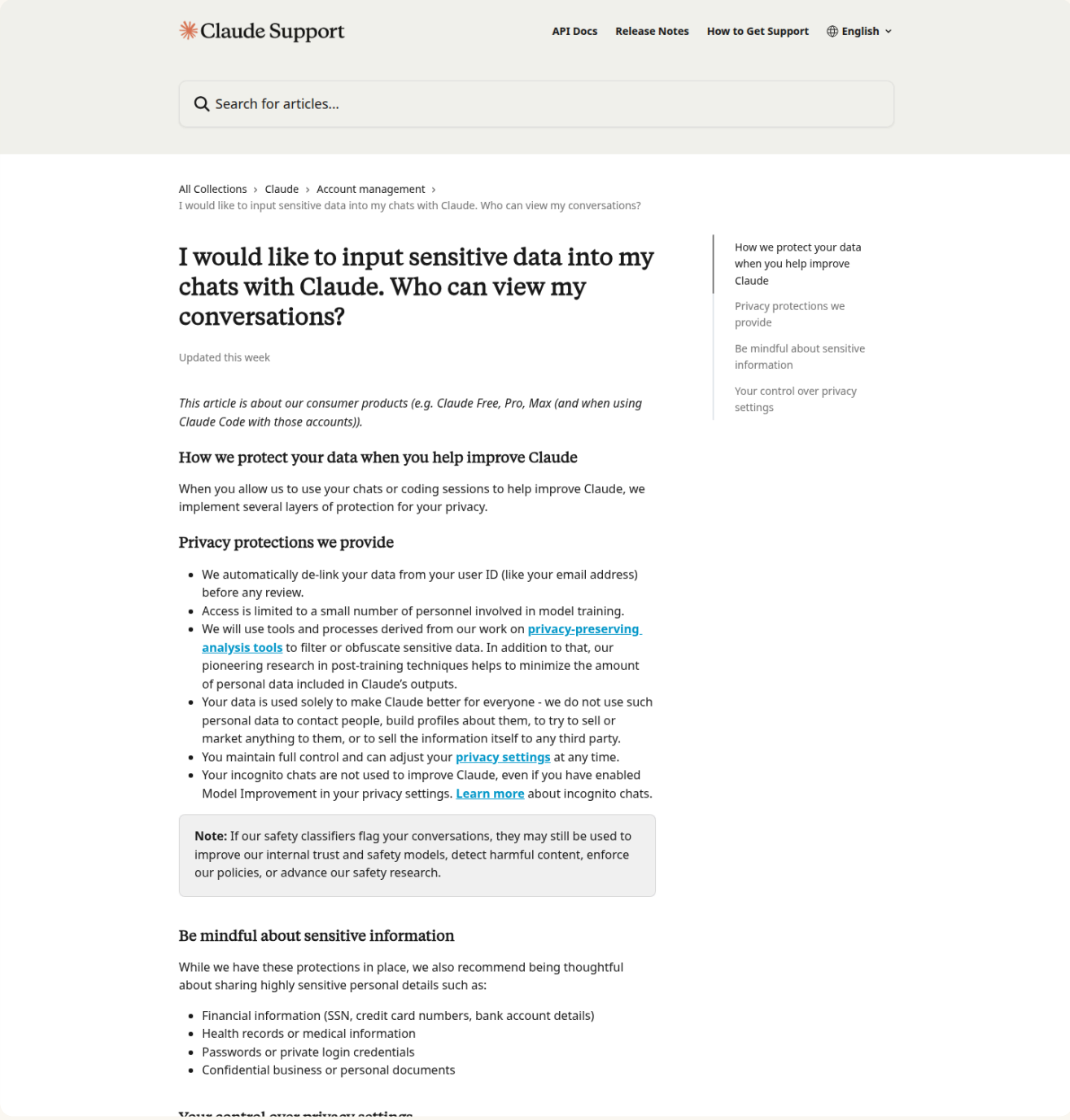
ไม่ใช่บทนโยบายบริษัท

แต่เป็น 5 กติกากันพลาดที่คนทำงานทั่วไปควรจำ

เพราะระบบนี้มีประโยชน์มาก

และเพราะมันมีประโยชน์มาก เราจึงต้องใช้ให้รอบคอบ

กติกาที่ 1: อย่าส่งข้อมูลลับถ้าไม่จำเป็น



หน้า Claude Support เรื่อง sensitive data: เตือนให้ตัดข้อมูลลับหรือข้อมูลส่วนตัวที่ไม่จำเป็นออกก่อน

ภาพ: หน้า Claude Support เรื่อง sensitive data: เตือนให้ตัดข้อมูลลับหรือข้อมูลส่วนตัวที่ไม่จำเป็นออกก่อน

Claude ช่วยอ่านไฟล์ ช่วยสรุป ช่วยจัด note ได้ดี

แต่ไม่ได้แปลว่าเราควรส่งทุกอย่างให้ Claude

ก่อน copy ข้อความหรือ upload file ให้ถามตัวเองว่า:

> ข้อมูลนี้จำเป็นต่อคำตอบไหม?
> ถ้าเอาชื่อลูกค้าออก ยังตอบได้ไหม?
> ถ้าเอาเบอร์โทร email เลขสัญญาออก ยังตอบได้ไหม?
> ถ้าคนอื่นในบริษัทมาเห็น prompt นี้ จะมีปัญหาไหม?

ถ้าคำตอบคือ “ไม่จำเป็น” ให้ตัดออกก่อน

ตัวอย่างข้อมูลที่ควรระวัง:

- ชื่อลูกค้าจริง
- เบอร์โทร email ที่อยู่
- ราคาเฉพาะราย
- สัญญา ใบเสนอราคา invoice
- ข้อมูลพนักงาน
- ข้อมูลสุขภาพ การเงิน หรือเรื่องส่วนตัว
- เอกสารที่บริษัทระบุว่าห้ามแชร์ออกนอกระบบ

ไม่ได้แปลว่าห้ามใช้ AI กับงานจริง

แต่ให้ใช้แบบลดข้อมูลเท่าที่ทำได้

ตัวอย่าง prompt ที่ดีกว่า:

> นี่คือคำถามจากลูกค้า B2B รายหนึ่ง ฉันลบชื่อบริษัทและข้อมูลส่วนตัวออกแล้ว
> ช่วยสรุป pain point และเสนอ FAQ ที่ควรตอบบนหน้าเว็บ

แทนที่จะส่ง email ลูกค้าทั้งฉบับพร้อมลายเซ็น เบอร์โทร และชื่อบริษัท

กติกาง่าย ๆ:

> ส่งเท่าที่จำเป็น ไม่ส่งเพื่อความสะดวกอย่างเดียว

ถ้าทำงานในบริษัท ให้ดูนโยบายของบริษัทก่อนเสมอ

ถ้าไม่แน่ใจ ให้ถามหัวหน้า ทีม IT หรือคนที่รับผิดชอบเรื่องเครื่องมือ AI

อย่าเดาเอง

กติกาที่ 2: อย่าเชื่อ Claude ถ้าไม่มี source

Claude เขียนได้ลื่น

และเพราะมันเขียนได้ลื่น เราจึงเผลอเชื่อง่าย

นี่คือจุดที่ต้องระวัง

Claude ช่วยสรุป ช่วยจัดโครง ช่วยเสนอไอเดียได้

แต่ Claude ไม่ใช่เจ้าของความจริง

ความจริงของ Wiki คุณควรมาจาก:

- source note
- meeting note
- email จริง
- เอกสารบริษัท
- feedback ลูกค้า
- decision ที่ทีมตกลงแล้ว
- link หรือไฟล์ที่อ้างอิงได้

เวลา Claude ตอบ ให้ถามเสมอว่า:

> คำตอบนี้อิงจาก source ไหน?
> มีส่วนไหนเป็น inference?
> มีส่วนไหนเป็น idea?
> มีส่วนไหน Claude แต่งเติมเพื่อให้ดูสมบูรณ์ไหม?

ถ้า Claude เขียนว่า:

> ลูกค้าส่วนใหญ่กังวลว่า AI training ต้องเขียนโค้ด

ให้ถามกลับว่า:

> จาก source ที่ให้มา คำว่า "ส่วนใหญ่" มีหลักฐานไหม
> ถ้าไม่มี ให้แก้เป็นภาษาที่ไม่เกินข้อมูล เช่น "ลูกค้าบางส่วน" หรือ "ลูกค้าถามบ่อยตาม meeting note นี้"

ภาษาที่ต่างกันเล็กน้อย อาจทำให้งานต่างกันมาก

“ลูกค้าส่วนใหญ่” = ฟังเหมือนมีข้อมูลเชิงสถิติ

“ลูกค้าบางส่วน” = ระวังกว่า

“จาก meeting ทีม sales ครั้งนี้ มีคำถามว่า...” = อิง source ชัดกว่า

กติกาง่าย ๆ:

> ถ้าไม่มี source ให้ถือว่าเป็น draft ไม่ใช่ความจริง

กติกาที่ 3: แยก Fact / Inference / Idea / Decision เสมอ

นี่คือกติกาที่ช่วยกันพลาดมากที่สุดในเล่ม

ให้แยก 4 อย่างนี้ให้ชัด

```
Fact      = สิ่งที่ source บอกจริง
Inference = สิ่งที่เราตีความจาก fact
Idea      = สิ่งที่อาจลองทำ
Decision  = สิ่งที่ตกลงแล้วว่าจะทำหรือไม่ทำ
```

ตัวอย่างจาก meeting ทีม sales:

> ประชุมทีม sales 25 พ.ค.
> ลูกค้าถามบ่อยว่า AI training ต้องรู้โค้ดไหม
> หลายคนกลัวทีมใช้ไม่เป็น
> ทีม sales อยากได้ FAQ หน้าเว็บ
> ควรมี webinar สำหรับผู้บริหารที่ไม่รู้ technical
> ต้องส่ง draft FAQ ภายในศุกร์นี้

แยกได้แบบนี้:

```markdown
## Fact
- ลูกค้าถามบ่อยว่า AI training ต้องรู้โค้ดไหม
- ลูกค้าหลายคนกลัวทีมใช้ไม่เป็น
- ทีม sales อยากได้ FAQ หน้าเว็บ
- ต้องส่ง draft FAQ ภายในศุกร์นี้

## Inference
- ลูกค้าอาจเข้าใจว่า AI training = programming
- หน้าเว็บควรสื่อสารชัดขึ้นว่าเหมาะกับทีม business

## Idea
- อาจทำ webinar สำหรับผู้บริหารที่ไม่ technical

## Decision
- ยังไม่มี decision ชัดเจนเรื่อง webinar
```

ถ้าคุณไม่แยก 4 อย่างนี้ ปัญหาจะเกิดเร็วมาก

Idea จะถูกพูดเหมือน decision

Inference จะถูกเขียนเหมือน fact

Draft จะถูกส่งเหมือน final

และ Claude จะยิ่งขยายความเข้าใจผิดให้ดูน่าเชื่อขึ้น

กติกาง่าย ๆ:

> ก่อน copy งานจาก Claude ไปใช้ ให้ถามว่าแต่ละประโยคเป็น Fact, Inference, Idea หรือ Decision

กติกาที่ 4: อย่าให้ระบบใหญ่กว่างาน

Obsidian ทำได้เยอะ

Claude ทำได้เยอะ

Plugin ก็มีเยอะ

Template ก็ทำได้ละเอียดมาก

แต่เล่มนี้ไม่ได้สอนให้คุณสร้างระบบที่สมบูรณ์แบบ

เล่มนี้สอนให้คุณกลับมาใช้ความรู้ของตัวเองได้เร็วขึ้น

ถ้าวันหนึ่งคุณพบว่า:

- ใช้เวลาตั้งชื่อ note นานกว่าทำงานจริง
- จัด folder มากกว่าสร้าง output
- แก้ template ทุกวันแต่ไม่เขียน note
- ติด plugin ใหม่เรื่อย ๆ แต่ Wiki ไม่ได้ดีขึ้น
- review นานจนไม่อยากเปิด Obsidian

แปลว่าระบบเริ่มใหญ่กว่างาน

ให้กลับไปใช้กติกาเริ่มต้น:

```
1 source → 1 source note
ถ้าใช้ซ้ำจริง → เพิ่ม concept note
ถ้ามีงานต้องทำ → เพิ่ม project note
ถ้ามีการตัดสินใจชัด → เพิ่ม decision note
```

แค่นี้พอ

ไม่ต้องมีระบบซับซ้อนตั้งแต่แรก

ไม่ต้องมี tag ครบทุกมุม

ไม่ต้องทำ graph ให้สวย

ไม่ต้อง import ชีวิตทั้งชีวิตเข้า vault

กติกาง่าย ๆ:

> ระบบที่ดีคือระบบที่คุณยังใช้ได้ในวันที่งานยุ่ง

ถ้าวันยุ่งแล้วใช้ไม่ได้ แปลว่ามันไม่ใช่ระบบของคุณ

มันเป็นงานอดิเรกอีกชิ้นหนึ่ง

กติกาที่ 5: ทบทวนเป็นรอบ ไม่ใช่จัดครั้งเดียวแล้วจบ

Wiki ไม่ใช่เอกสารที่เขียนเสร็จแล้ววางไว้

Wiki เป็นของที่โตไปพร้อมงาน

ดังนั้นอย่าคาดหวังว่าตั้งครั้งแรกแล้วจะถูกหมด

ชื่อ note บางอันจะเปลี่ยน

folder บางอันจะไม่พอดี

concept บางหน้าจะรวมกัน

project บางอันจะถูกปิด

Home จะต้องจัดใหม่

นี่เป็นเรื่องปกติ

ให้ใช้รอบทบทวนง่าย ๆ:

> ทุกวันหรือเมื่องานเข้า:
> - Capture source เข้า inbox
>
> สัปดาห์ละครั้ง:
> - ทำ Weekly Review 30 นาที
>
> เดือนละครั้ง:
> - ดูว่า Wiki ยังช่วยงานหลักอยู่ไหม

ไม่ต้องดูแล Wiki ทั้งวัน

แค่มีจังหวะกลับมาจัดบ้าน

ถ้าไม่ทบทวน Wiki จะค่อย ๆ กลับไปเป็นกองข้อมูลเหมือนเดิม

ถ้าทบทวนบ้าง Wiki จะกลายเป็นสินทรัพย์ของงาน

กติกาง่าย ๆ:

> Wiki ที่ดีไม่ได้เกิดจากการจัดครั้งใหญ่ แต่เกิดจากการจัดเล็ก ๆ ซ้ำ ๆ

Checklist: ก่อนส่งงานที่ใช้ Claude ช่วย

ก่อนส่งข้อมูลให้ Claude ให้ผ่านด่านสั้น ๆ ว่าจำเป็น ปลอดภัย และมี source พอหรือยัง

ภาพ: ก่อนส่งข้อมูลให้ Claude ให้ผ่านด่านสั้น ๆ ว่าจำเป็น ปลอดภัย และมี source พอหรือยัง

ก่อนเอางานจาก Claude ไปส่งหัวหน้า ลูกค้า หรือทีม ใช้ checklist นี้

```
[ ] มี source รองรับส่วนสำคัญไหม
[ ] Claude แต่งข้อมูลเพิ่มหรือเปล่า
[ ] แยก fact / inference / idea / decision แล้วหรือยัง
[ ] มีข้อมูลลับหรือข้อมูลส่วนตัวติดไปไหม
[ ] ตัวเลข วันที่ ชื่อคน ชื่อลูกค้า ถูกต้องไหม
[ ] ภาษาสอดคล้องกับบริบทบริษัทไหม
[ ] ถ้าเป็น decision ตรวจแล้วหรือยังว่าตัดสินใจจริง
[ ] ถ้าเป็นคำแนะนำ มีใครควร review ก่อนส่งไหม
```

ถ้าไม่ครบทุกข้อ ไม่ได้แปลว่าห้ามส่ง

แต่แปลว่าคุณควรรู้ว่าจุดไหนยังเสี่ยง

Checklist: ก่อนใช้ Wiki กับทีม

ถ้าจะชวนทีมใช้ Wiki ด้วยกัน อย่าเริ่มจากการสอนทุกฟีเจอร์

เริ่มจากกติกาง่าย ๆ

```
[ ] ทุก source สำคัญต้องมีที่มา
[ ] ทุก project สำคัญต้องมี project note
[ ] ทุก decision สำคัญต้องบอกว่าใคร/เมื่อไหร่/ตัดสินใจอะไร
[ ] ใช้ชื่อ note ที่คนในทีมอ่านแล้วเข้าใจ
[ ] ไม่ใส่ข้อมูลลับเกินจำเป็น
[ ] มี Home หรือ Index ที่ทีมเริ่มอ่านได้
[ ] มีคนรับผิดชอบ Weekly Review
[ ] Claude ช่วย draft/review ได้ แต่คนในทีมต้องตรวจสุดท้าย
```

ระบบทีมไม่ต้องเริ่มใหญ่

เริ่มจาก project เดียวก็พอ

เช่น:

```
FAQ หน้าเว็บคอร์ส AI
```

ให้ทีมเห็นก่อนว่า Wiki ช่วยงานจริงยังไง

แล้วค่อยขยาย

สรุปทั้งเล่มในภาพเดียว

ถ้าต้องจำทั้งเล่มให้เหลือภาพเดียว ให้จำแบบนี้:

```
Obsidian = บ้านของความรู้
Claude = ผู้ช่วยอ่านและจัดบ้าน
คุณ = เจ้าของบ้านและ editor สุดท้าย
```

Obsidian เก็บไฟล์ของคุณให้อยู่กับคุณ

Claude ช่วยอ่าน ช่วยสรุป ช่วยถาม ช่วยร่าง

LLM Wiki ทำให้ความรู้มี source, link, index และรูปแบบที่ AI เข้าใจง่ายขึ้น

แต่สุดท้าย คนที่ต้องตัดสินใจคือคุณ

คุณเลือกว่าจะเก็บอะไร

คุณเลือกว่าจะเชื่ออะไร

คุณเลือกว่าจะส่งอะไรออกไป

คุณเลือกว่าจะทำให้ระบบเล็กพอที่จะใช้ต่อได้จริง

จบเล่มหลัก: เริ่มจากเล็กที่สุด

อย่าเริ่มจากการย้ายทุกอย่างเข้า Obsidian

อย่าเริ่มจาก plugin

อย่าเริ่มจากการสร้างระบบสมบูรณ์แบบ

เริ่มจาก source หนึ่งชิ้น

เช่น meeting note ล่าสุด

ทำแบบนี้:

> 1. ใส่เข้า 00-inbox
> 2. เพิ่ม date/source/context
> 3. ให้ Claude ช่วยแยก fact/inference/idea
> 4. สร้าง source note
> 5. ถ้ามีเรื่องใช้ซ้ำ สร้าง concept note
> 6. ถ้ามีงานต้องทำ สร้าง project note
> 7. link กลับไปมา
> 8. อัปเดต Home และ Log
> 9. สัปดาห์หน้ากลับมา review

ถ้าทำได้หนึ่งครั้ง คุณเริ่มใช้ LLM Wiki แล้ว

ถ้าทำซ้ำได้ คุณจะมีคลังความรู้ที่ไม่ได้แค่เก็บข้อมูล

แต่ช่วยให้คุณคิดต่อ ทำงานต่อ และใช้ Claude ได้ดีขึ้นจริง

Artifact ท้ายบท: 5 กติกากันพลาด

> 1. อย่าส่งข้อมูลลับถ้าไม่จำเป็น
> 2. อย่าเชื่อ Claude ถ้าไม่มี source
> 3. แยก Fact / Inference / Idea / Decision เสมอ
> 4. อย่าให้ระบบใหญ่กว่างาน
> 5. ทบทวนเป็นรอบ ไม่ใช่จัดครั้งเดียวแล้วจบ

Artifact ท้ายบท: Prompt ตรวจงานก่อนส่ง

> ช่วยตรวจ draft นี้ก่อนส่งงานจริง
>
> สิ่งที่ต้องการ:
> 1. แยกประโยคสำคัญเป็น Fact / Inference / Idea / Decision
> 2. ชี้ว่าจุดไหนต้องมี source เพิ่ม
> 3. ชี้ว่าจุดไหนอาจเป็นการแต่งข้อมูลเกิน source
> 4. ชี้ว่ามีข้อมูลลับหรือข้อมูลส่วนตัวที่ควรลบไหม
> 5. เสนอ version ที่ระวังกว่า ถ้าภาษาปัจจุบันฟันธงเกินไป
>
> กติกา:
> - อย่าเพิ่มข้อมูลใหม่
> - ถ้าไม่มี source ให้บอกว่าไม่มี source
> - ใช้ภาษาคนทำงานทั่วไป

อ้างอิงหลักของบทนี้

- Claude Help Center: Upload files to Claude
- Claude Help Center: Projects and project knowledge
- Anthropic Support/Privacy resources ที่เกี่ยวกับการใช้ข้อมูลและการควบคุมข้อมูล

- Obsidian Help: Properties, Internal links, Tags และ Search
- Andrej Karpathy: LLM Wiki gist: source, lint, index/log pattern
- `AgriciDaniel/claude-obsidian`: LLM Wiki Pattern และ Source-First Synthesis

APPENDIX A

Prompt Library

ส่วนนี้ไม่ใช่บทอ่านยาว

เป็นกล่องเครื่องมือท้ายเล่ม

เปิดมาคัดลอกไปใช้ได้เลย

ใช้เมื่อคุณอยากเริ่มทำ LLM Wiki ใน Obsidian กับ Claude โดยไม่ต้องคิด format ใหม่ทุกครั้ง

ของใน Appendix แบ่งเป็น 5 ชุด:

1. Prompt Library
2. Starter Templates
3. Checklist รวม
4. ตัวอย่างการตั้งชื่อ note สำหรับคนไทย
5. แผนเริ่มใช้ใน 7 วัน

ถ้ายังไม่รู้จะเริ่มตรงไหน ให้เริ่มจากชุดนี้:

> 1. สร้าง vault ตาม checklist
> 2. ใช้ Source Template กับข้อมูลหนึ่งชิ้น
> 3. ใช้ Source-to-Wiki Prompt กับ Claude
> 4. สร้าง Project Note ถ้ามีงานต้องทำ
> 5. ทำ Weekly Review วันศุกร์

แค่นี้พอสำหรับ version แรก

A1: Setup Assistant Prompt

ใช้ตอนเริ่มสร้าง vault แรก

```
ฉันกำลังสร้าง Obsidian vault สำหรับทำ LLM Wiki ส่วนตัว/ทีมเล็ก
คนใช้ไม่ใช่ developer
เป้าหมายคือเก็บ source, note, project, decision และให้ Claude ช่วยอ่าน/จัดระบบได้ง่ายขึ้น

ช่วยออกแบบ setup เริ่มต้นให้ฉัน โดยใช้โครงสร้างนี้:
- 00-inbox
- 10-sources
- 20-notes
- 30-concepts
- 40-projects
- 90-index
- _templates
- Home.md
- Log.md

สิ่งที่ต้องการ:
1. อธิบายว่าแต่ละ folder ใช้ทำอะไรแบบสั้น ๆ
2. เสนอ Home.md เวอร์ชันแรก
3. เสนอ Log.md เวอร์ชันแรก
4. เสนอ template สำหรับ Source / Concept / Project / Decision
5. ให้ checklist 30 นาทีสำหรับตั้งค่า

กติกา:
- ไม่ต้องใช้ plugin
- ไม่ต้องใช้ code
- ใช้ภาษาคนทำงานทั่วไป
- ทำให้เริ่มใช้ได้วันนี้
```

A2: Source-to-Wiki Prompt

ใช้บ่อยที่สุด

ใช้เมื่อต้องการแปลง source หนึ่งชิ้นให้กลายเป็น note ที่ใช้ต่อได้

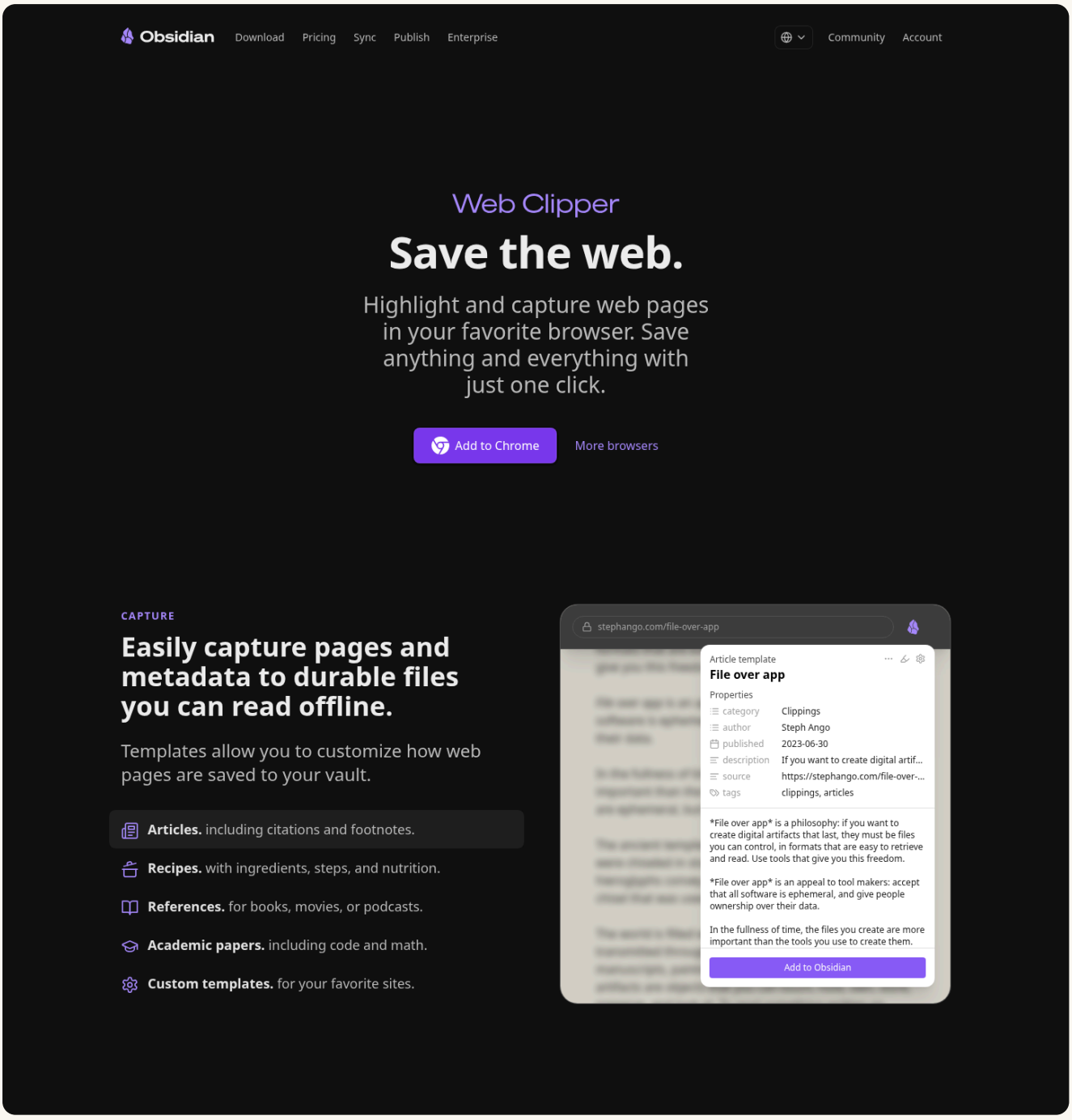
```
ฉันกำลังทำ LLM Wiki ใน Obsidian
ช่วยแปลง source นี้เป็น wiki notes ที่ใช้ต่อได้จริง

ข้อมูล context:
- งาน/โปรเจกต์ที่เกี่ยวข้อง:
- กลุ่มคนที่เกี่ยวข้อง:
- เป้าหมายของการเก็บ source นี้:

สิ่งที่ต้องการ:
1. Source note พร้อม Summary, Fact, Inference, Idea, Questions
2. Concept notes ที่ควรสร้างหรืออัปเดต
3. Project note ที่เกี่ยวข้อง ถ้ามี
4. Decision note ถ้ามี decision ชัดเจนเท่านั้น
5. Internal links ที่ควรเชื่อม
6. Home.md ควรเพิ่ม link อะไร
7. Log.md ควรบันทึกอะไร

กติกา:
- ใช้เฉพาะข้อมูลที่ฉันให้
- อย่าแต่งข้อมูลให้ดูสมบูรณ์
- ถ้าข้อมูลไม่พอ ให้เขียนว่า "ข้อมูลไม่พอ"
- แยก fact / inference / idea ให้ชัด
- ถ้าเป็นแค่ idea อย่าเขียนเป็น decision
- ใช้ชื่อ note ที่คนทำงานทั่วไปอ่านแล้วเข้าใจ
- เขียนเป็น Markdown ที่ copy ไปใช้ใน Obsidian ได้

Source:
[วาง source ที่นี่]
```

A3: Meeting Note to Wiki Prompt

ใช้กับบันทึกประชุม

```
ช่วยแปลง meeting note นี้เป็น LLM Wiki notes สำหรับ Obsidian

สิ่งที่ต้องการ:
1. สรุป meeting เป็น Source note
2. แยก Fact / Inference / Idea / Decision / Todo
3. เสนอ Project note ถ้ามีงานต้องทำต่อ
4. เสนอ Concept note ถ้ามีเรื่องที่ควรใช้ซ้ำ
5. เสนอคำถามที่ต้องถามคนจริงเพิ่ม
6. เสนอ internal links ที่ควรมี

กติกา:
- อย่าเดาชื่อคนหรือ owner ถ้าไม่มีใน note
- อย่าเปลี่ยน idea ให้เป็น decision
- ถ้า deadline ไม่ชัด ให้เขียนว่าไม่ชัด
- ใช้ Markdown ที่ copy เข้า Obsidian ได้

Meeting note:
[วางบันทึกประชุมที่นี่]
```

A4: Article / Webpage to Wiki Prompt



Web Clipper ช่วยเก็บหน้าเว็บเป็น source ได้ แต่ workflow สำคัญกว่าการติดเครื่องมือเพิ่ม

ภาพ: Web Clipper ช่วยเก็บหน้าเว็บเป็น source ได้ แต่ workflow สำคัญกว่าการติดเครื่องมือเพิ่ม

ใช้กับบทความ หน้าเว็บ หรือ research

```
ฉันกำลังเก็บบทความ/หน้าเว็บนี้เข้า LLM Wiki ใน Obsidian
ช่วยสรุปให้เป็น note ที่ใช้ต่อได้

สิ่งที่ต้องการ:
1. Source note พร้อม summary และ key claims
2. แยก fact / claim / inference / idea
3. บอกว่าเนื้อหานี้เกี่ยวกับ project หรือ concept ไหนใน Wiki
4. เสนอ concept note ที่ควรสร้างหรืออัปเดต
5. เสนอคำถามที่ควรตรวจเพิ่มก่อนนำไปใช้จริง
6. เสนอ internal links

กติกา:
- อย่าคัดลอกบทความยาว ๆ
- อย่าเขียนเกิน source
- ถ้า claim ไม่มีหลักฐานในข้อความที่ให้ ให้บอกว่าไม่มีหลักฐานใน source นี้
- แยก observation จากข้อเสนอของเรา

Context งานของฉัน:
[อธิบายสั้น ๆ]

เนื้อหา/source:
[วางบทความหรือ excerpt ที่นี่]
```

A5: Customer Email to Wiki Prompt

ใช้กับ email ลูกค้าหรือข้อความจากลูกค้า

ก่อนใช้ prompt นี้ ให้ลบข้อมูลส่วนตัวที่ไม่จำเป็นก่อน

```
นี่คือข้อความจากลูกค้า ฉันลบข้อมูลส่วนตัวที่ไม่จำเป็นออกแล้ว
ช่วยแปลงเป็น wiki note สำหรับ Obsidian

สิ่งที่ต้องการ:
1. สรุปสิ่งที่ลูกค้าถามหรือกังวล
2. แยก Fact / Inference / Idea / Todo
3. เสนอว่าควรสร้างหรืออัปเดต Concept note อะไร
4. เสนอว่าข้อความนี้เกี่ยวกับ Project note ไหน
5. เสนอ FAQ หรือ response idea ถ้ามี
6. ระบุข้อมูลที่ยังต้องถามลูกค้าเพิ่ม

กติกา:
- อย่าเดาข้อมูลลูกค้าเพิ่ม
- อย่าเขียนตอบในนามบริษัทแบบ final
- ถ้าข้อมูลไม่พอ ให้บอกว่าไม่พอ
- ระวังข้อมูล sensitive

ข้อความลูกค้า:
[วางข้อความที่ลบข้อมูลส่วนตัวแล้ว]
```

A6: Ask Claude to Find Missing Questions

ใช้เมื่อมี note แล้ว แต่อยากรู้ว่ายังขาดอะไร

```
ช่วยอ่าน note นี้แล้วหาช่องว่างของข้อมูล

สิ่งที่ต้องการ:
1. คำถามที่ควรถามคนจริงเพิ่ม
2. จุดที่ source ยังไม่พอ
3. จุดที่เป็น inference แต่ยังไม่มี fact รองรับ
4. จุดที่อาจสับสนระหว่าง idea กับ decision
5. สิ่งที่ควรอัปเดตใน project note หรือ concept note

กติกา:
- อย่าเสนอคำตอบใหม่ถ้าไม่มี source
- ให้เน้นคำถามที่ช่วยทำงานต่อได้จริง
- เขียนเป็น checklist สั้น ๆ

Note:
[วาง note ที่นี่]
```

A7: Weekly Review Prompt

ใช้สัปดาห์ละครั้งเพื่อไม่ให้ Wiki รก

```
ฉันกำลังทำ Weekly Review ของ LLM Wiki ใน Obsidian
เป้าหมายคือทำให้ Wiki ไม่รก หาเจอ และใช้ต่อกับงานจริงได้

ข้อมูลที่ฉันให้:
- Home.md ปัจจุบัน
- Log.md ล่าสุด
- รายชื่อ note ใน 00-inbox
- project notes ที่ active/waiting
- รายชื่อ concept notes บางส่วน

ช่วย review 6 เรื่อง:
1. Inbox: note ไหนควร process / keep raw / delete / ask later
2. Projects: project ไหนควร active / waiting / done / paused / archived
3. Duplicates: note ไหนอาจซ้ำหรือควรรวม
4. Links: note ไหนดูเหมือนขาด link สำคัญ
5. Home/Index: Home ควรปรับอะไร และควรสร้าง index note อะไร
6. Questions: มีคำถามอะไรที่ต้องถามคนจริงก่อนตัดสินใจ

กติกา:
- อย่าเดาข้อมูลที่ไม่มีใน note
- ถ้าไม่แน่ใจ ให้เขียนว่า "ต้องตรวจเอง"
- อย่าเสนอให้ลบอะไรถ้าเหตุผลไม่ชัด
- แยก fact / inference / suggestion ให้ชัด
- เขียนเป็น checklist ที่ฉันทำตามได้ใน 30 นาที
```

A8: Draft Check Before Sending Prompt

ใช้ก่อนส่งงานให้หัวหน้า ลูกค้า หรือทีม

```
ช่วยตรวจ draft นี้ก่อนส่งงานจริง

สิ่งที่ต้องการ:
1. แยกประโยคสำคัญเป็น Fact / Inference / Idea / Decision
2. ชี้ว่าจุดไหนต้องมี source เพิ่ม
3. ชี้ว่าจุดไหนอาจเป็นการแต่งข้อมูลเกิน source
4. ชี้ว่ามีข้อมูลลับหรือข้อมูลส่วนตัวที่ควรลบไหม
5. เสนอ version ที่ระวังกว่า ถ้าภาษาปัจจุบันฟันธงเกินไป

กติกา:
- อย่าเพิ่มข้อมูลใหม่
- ถ้าไม่มี source ให้บอกว่าไม่มี source
- ใช้ภาษาคนทำงานทั่วไป

Draft:
[วาง draft ที่นี่]
```

A9: Home Cleanup Prompt

ใช้เมื่อ Home เริ่มยาวเกินไป

```
นี่คือ Home.md ปัจจุบันของ LLM Wiki และรายชื่อ note สำคัญบางส่วน
ช่วยเสนอวิธีจัด Home ให้สั้นและใช้เริ่มงานได้ดีขึ้น

สิ่งที่ต้องการ:
1. ส่วนไหนควรอยู่ใน Home ต่อ
2. ส่วนไหนควรย้ายไป index แยก
3. ควรสร้าง index note อะไรบ้าง
4. ตัวอย่าง Home.md เวอร์ชันใหม่แบบสั้น

กติกา:
- Home ต้องอ่านจบใน 1 นาที
- อย่าใส่ทุก note ลง Home
- ใช้ภาษาคนทำงานทั่วไป
- ถ้าไม่แน่ใจ ให้เสนอเป็น option ไม่ฟันธง

Home ปัจจุบัน:
[วาง Home.md]

รายชื่อ note สำคัญ:
[วางรายชื่อ note]
```

A10: Duplicate Notes Prompt

ใช้เมื่อมี note หลายชื่อแต่เรื่องคล้ายกัน

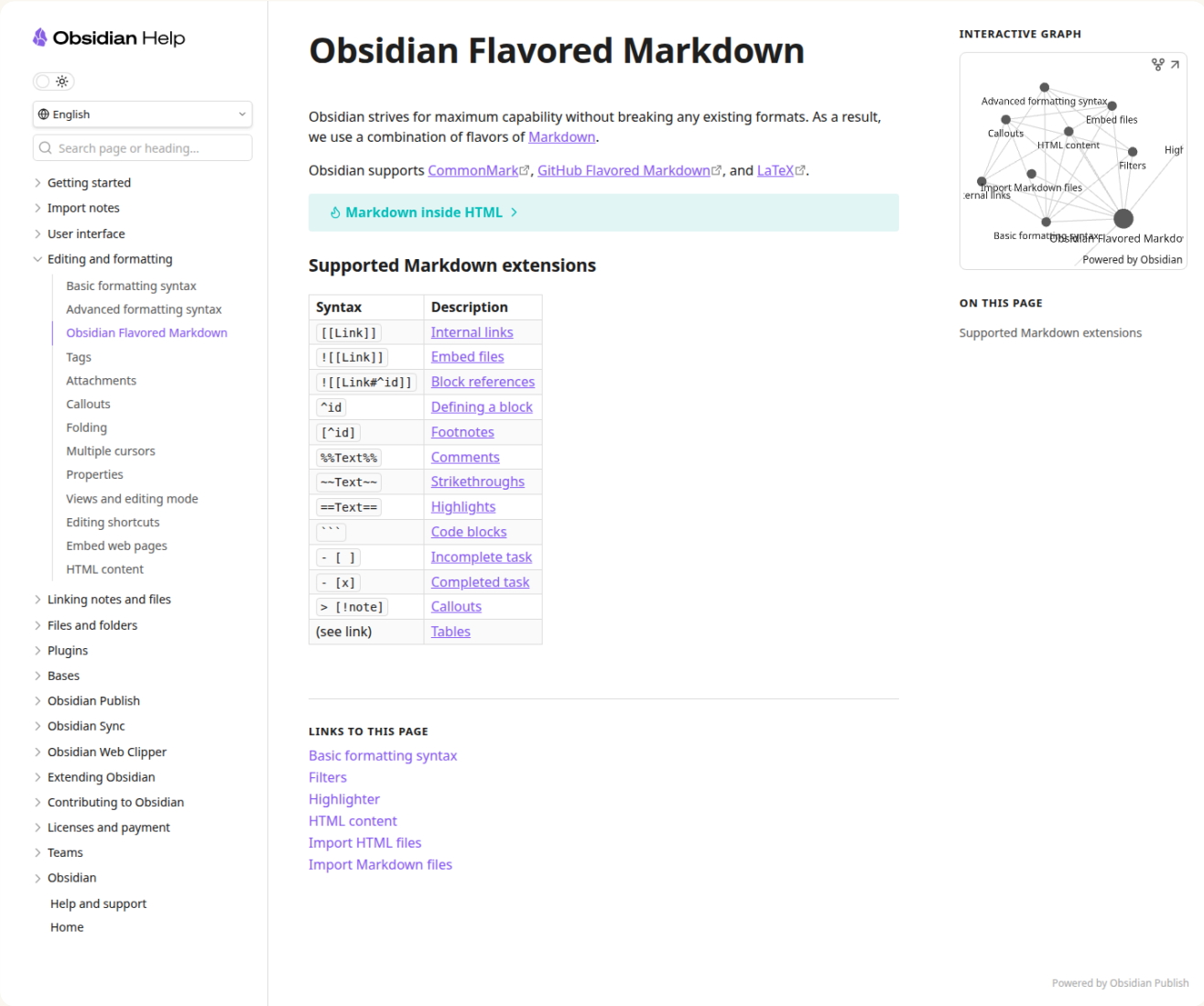
นี่คือรายชื่อ note ใน Obsidian ของฉัน
ช่วยหากลุ่ม note ที่อาจซ้ำหรือใกล้กันเกินไป

สิ่งที่ต้องการ:
1. จัดกลุ่ม note ที่ชื่อคล้ายกันหรือพูดเรื่องใกล้กัน
2. เสนอว่า note ไหนควรเป็นหน้าหลัก
3. เสนอ note ไหนควรรวม/เปลี่ยนเป็นทางเข้า/ปล่อยไว้
4. ระบุว่าข้อเสนอไหนต้องเปิดเนื้อหาดูเพิ่มก่อน

กติกา:
- อย่าฟันธงจากชื่ออย่างเดียวถ้าไม่พอ
- ใช้คำว่า "อาจซ้ำ" ไม่ใช่ "ซ้ำแน่นอน"
- เป้าหมายคือทำให้ Wiki ใช้ง่ายขึ้น ไม่ใช่ลบให้เหลือน้อยที่สุด

รายชื่อ note:
[วางรายชื่อ note]

Starter Templates



Markdown คือรูปแบบข้อความธรรมดาที่ทำหัวข้อ list และ link ได้ เหมาะกับ template ใน Appendix

ภาพ: Markdown คือรูปแบบข้อความธรรมดาที่ทำหัวข้อ list และ link ได้ เหมาะกับ template ใน Appendix

คัดลอก template เหล่านี้ไปไว้ใน folder `_templates`

ถ้ายังไม่ใช้ระบบ template ใน Obsidian ก็ไม่เป็นไร

สร้าง note ใหม่แล้ว copy ข้อความไปวางเองได้

B1: Source Note Template

MARKDOWN

```
# {{title}}

type: source
status: raw
date: YYYY-MM-DD
source:

## Context
source นี้คืออะไร และเก็บไว้ทำไม

## Raw / Original
วางข้อมูลต้นทางหรือสรุปต้นทางที่ยังไม่จัดระบบ

## Summary
สรุปสั้น ๆ 3-5 บรรทัด

## Fact
-

## Inference
-

## Idea
-

## Useful for
- [[Project หรือ Concept ที่เกี่ยวข้อง]]

## Questions
-

## Links
-
```

B2: Concept Note Template

MARKDOWN

```
# {{title}}

type: concept
status: draft
review: YYYY-MM-DD

## ความหมายสั้น ๆ
อธิบาย concept นี้แบบคนทำงานทั่วไปเข้าใจ

## ทำไมเรื่องนี้สำคัญ
-

## ใช้กับงานไหน
-

## ตัวอย่าง
-

## Sources
- [[Source note]]

## Related
- [[Concept หรือ Project ที่เกี่ยวข้อง]]

## Questions
-
```

B3: Project Note Template

MARKDOWN

```
# {{title}}

type: project
status: draft
owner:
deadline:
review: YYYY-MM-DD

## เป้าหมาย
project นี้ต้องการ output อะไร

## Context สำคัญ
-

## Sources
- [[Source note]]

## Concepts
- [[Concept note]]

## Decisions
- [[Decision note]]

## Todo
- [ ]

## Draft / Working area
พื้นที่ร่างงานหรือรวบรวม idea

## Questions
-

## Next review
YYYY-MM-DD
```

B4: Decision Note Template

MARKDOWN

```
# {{title}}

type: decision
status: active
date: YYYY-MM-DD
decided-by:
related-project:

## Decision
ตัดสินใจว่าอะไร

## Context
ทำไมต้องตัดสินใจเรื่องนี้

## Options considered
- Option 1:
- Option 2:

## Reason
เหตุผลที่เลือกทางนี้

## Impact
สิ่งที่ต้องทำต่อ หรือสิ่งที่เปลี่ยนจาก decision นี้

## Sources
- [[Source note]]
- [[Meeting note]]

## Review / Expiry
ควรกลับมาทบทวนเมื่อไหร่
```

B5: Home.md Template

MARKDOWN

```
# Home

นี่คือหน้าเริ่มต้นของ LLM Wiki

## Start here
- [[Log]]
- [[Active Projects]]

## Projects สำคัญตอนนี้
-

## Concepts สำคัญ
-

## Sources ล่าสุดที่ควรรู้
-

## Index
- [[Active Projects]]
- [[Key Concepts]]
- [[Customer Questions]]
```

B6: Log.md Template

MARKDOWN

```
# Log

ใช้บันทึกสิ่งที่เปลี่ยนใน Wiki แบบสั้น ๆ

## YYYY-MM-DD

### Processed
-

### Created / Updated
-

### Decisions
-

### Open questions
-

### Next
-
```

B7: Weekly Review Note Template

MARKDOWN

```
# Weekly Review YYYY-MM-DD

type: review
status: done
date: YYYY-MM-DD

## Inbox
- Process now:
- Keep raw:
- Delete:
- Ask later:

## Projects
- Active:
- Waiting:
- Done:
- Paused / Archived:

## Duplicates / Merge candidates
-

## Missing links
-

## Home / Index updates
-

## Open questions
-

## Next week
- [ ]
```

B8: Review Later Template

MARKDOWN

```
# Review Later

ใช้เก็บสิ่งที่ยังไม่อยากตัดสินใจตอนนี้

## ต้องดูต่อ
-

## อาจซ้ำ
-

## อาจ archive
-

## ต้องถามคนอื่น
-
```

Checklist รวม

C1: 30-Minute Setup Checklist

```
[ ] สร้าง vault ใหม่ เช่น my-wiki
[ ] สร้าง folder: 00-inbox
[ ] สร้าง folder: 10-sources
[ ] สร้าง folder: 20-notes
[ ] สร้าง folder: 30-concepts
[ ] สร้าง folder: 40-projects
[ ] สร้าง folder: 90-index
[ ] สร้าง folder: _templates
[ ] สร้าง Home.md
[ ] สร้าง Log.md
[ ] สร้าง template Source / Concept / Project / Decision
[ ] เอา source จริงหนึ่งชิ้นเข้า 00-inbox
[ ] ใช้ Claude ช่วยแปลง source เป็น note แรก
[ ] อัปเดต Home และ Log
```

C2: Source Processing Checklist

```
[ ] Source มีวันที่และที่มา
[ ] เพิ่ม context สั้น ๆ แล้ว
[ ] ลบข้อมูลลับ/ข้อมูลส่วนตัวที่ไม่จำเป็นแล้ว
[ ] ส่งให้ Claude ด้วย prompt ที่ชัด
[ ] ตรวจว่า Claude ไม่แต่งข้อมูลเพิ่ม
[ ] แยก Fact / Inference / Idea / Decision แล้ว
[ ] สร้าง note เท่าที่จำเป็น
[ ] เชื่อม link สำคัญ
[ ] อัปเดต Home หรือ Index ถ้าควร
[ ] อัปเดต Log
[ ] จด open questions ถ้าข้อมูลยังขาด
```

C3: Weekly Review Checklist

```
[ ] เปิด 00-inbox
[ ] จัด note เป็น Process now / Keep raw / Delete / Ask later
[ ] เปิด project ที่ status active/waiting
[ ] อัปเดต status, deadline, next action
[ ] หา note ที่ชื่อหรือเนื้อหาใกล้กันเกินไป
[ ] เลือก note หลักถ้ามีเรื่องซ้ำ
[ ] หา note สำคัญที่ไม่มี link
[ ] เพิ่ม link เฉพาะที่ช่วยทำงานต่อ
[ ] ทำ Home ให้สั้นและเริ่มงานได้เร็ว
[ ] สร้าง index ย่อยถ้า Home ยาวเกินไป
[ ] เขียน Weekly Log
[ ] จดคำถามที่ต้องถามคนจริง
```

C4: Before Sending Work Checklist

```
[ ] มี source รองรับส่วนสำคัญไหม
[ ] Claude แต่งข้อมูลเพิ่มหรือเปล่า
[ ] แยก Fact / Inference / Idea / Decision แล้วหรือยัง
[ ] มีข้อมูลลับหรือข้อมูลส่วนตัวติดไปไหม
[ ] ตัวเลข วันที่ ชื่อคน ชื่อลูกค้า ถูกต้องไหม
[ ] ภาษาสอดคล้องกับบริบทบริษัทไหม
[ ] ถ้าเป็น decision ตรวจแล้วหรือยังว่าตัดสินใจจริง
[ ] ถ้าเป็นคำแนะนำ มีใครควร review ก่อนส่งไหม
```

C5: Team Wiki Checklist

```
[ ] ทุก source สำคัญต้องมีที่มา
[ ] ทุก project สำคัญต้องมี project note
[ ] ทุก decision สำคัญต้องบอกว่าใคร/เมื่อไหร่/ตัดสินใจอะไร
[ ] ใช้ชื่อ note ที่คนในทีมอ่านแล้วเข้าใจ
[ ] ไม่ใส่ข้อมูลลับเกินจำเป็น
[ ] มี Home หรือ Index ที่ทีมเริ่มอ่านได้
[ ] มีคนรับผิดชอบ Weekly Review
[ ] Claude ช่วย draft/review ได้ แต่คนในทีมต้องตรวจสุดท้าย
```

APPENDIX D

ตัวอย่างการตั้งชื่อ note สำหรับคนไทย

D1: หลักการตั้งชื่อ

ชื่อ note ควรทำ 3 อย่าง:

1. อ่านแล้วรู้ว่าเรื่องอะไร
2. search เจอด้วยคำที่คนใช้จริง
3. ไม่ยาวจนกลายเป็นประโยคทั้งย่อหน้า

กติกาง่าย ๆ:

> ชื่อ note = เรื่องหลัก + บริบทที่จำเป็น

D2: Source note

ใช้ pattern:

> [ชนิด source] + [เรื่อง] + [วันที่ ถ้าจำเป็น]

ตัวอย่าง:

```
Meeting ทีม sales เรื่องคำถามลูกค้า AI training.md
Email ลูกค้าเรื่องราคา AI training 2026-05-25.md
บทความ AI adoption สำหรับ SME.md
Transcript call ลูกค้าเรื่อง onboarding.md
หน้าเว็บคู่แข่งเรื่องคอร์ส AI สำหรับผู้บริหาร.md
```

อย่าตั้งชื่อแบบนี้:

```
note1.md
ประชุม.md
AI.md
สำคัญมาก.md
```

เพราะอีกสองสัปดาห์จะหาไม่เจอ

D3: Concept note

ใช้ชื่อเป็น “แนวคิดกลาง” ไม่ใช่ชื่อ meeting

ตัวอย่าง:

```
AI training สำหรับคนไม่รู้โค้ด.md
ความกังวลของลูกค้าต่อ AI training.md
FAQ สำหรับหน้าเว็บคอร์ส AI.md
ราคาและความคุ้มค่าของ AI training.md
การใช้ Claude กับงาน sales.md
```

Concept ควรใช้ซ้ำได้มากกว่าหนึ่ง project

ถ้าใช้ได้แค่ครั้งเดียว อาจยังไม่ต้องสร้าง concept note

D4: Project note

ใช้ pattern:

```
[output หรือ project] + [บริบท]
```

ตัวอย่าง:

```
FAQ หน้าเว็บคอร์ส AI.md
Webinar AI สำหรับผู้บริหาร.md
Sales script สำหรับคอร์ส AI.md
Landing page คอร์ส AI สำหรับทีม business.md
Internal training Claude สำหรับทีม marketing.md
```

Project note ควรมีงานต่อ มี todo หรือมี deadline

ถ้าไม่มีงานต่อ อาจเป็นแค่ idea ใน source note ก่อน

D5: Decision note

ใช้ pattern:

```
Decision - [ตัดสินใจเรื่องอะไร] - [วันที่]
```

ตัวอย่าง:

```
Decision - ใช้ FAQ เป็น content หลักหน้าเว็บ AI training - 2026-05-25.md
Decision - เลื่อน webinar ผู้บริหารเป็นเดือนกรกฎาคม - 2026-06-03.md
Decision - ไม่ใส่ราคาเต็มใน landing page - 2026-06-10.md
```

Decision note ต้องมีการตัดสินใจจริง

ถ้ายังเป็น “อาจทำ” ให้เก็บเป็น idea ก่อน

D6: Index note

ใช้ชื่อเป็นหมวดที่คนจะเริ่มอ่าน

ตัวอย่าง:

```
Active Projects.md
Key Concepts.md
Customer Questions.md
AI Training Content.md
Sales and Customer Questions.md
Claude Prompts.md
```

Index ไม่ต้องมีทุกอย่าง

มีไว้เป็นแผนที่ของเรื่องสำคัญ

APPENDIX E

แผนเริ่มใช้ใน 7 วัน

แผนนี้เหมาะกับคนที่อยากเริ่ม แต่ไม่อยากทำทีเดียวเยอะ

วันละ 15–30 นาทีพอ

Day 1: สร้างบ้าน

ทำแค่ setup

```
[ ] สร้าง vault
[ ] สร้าง folder หลัก
[ ] สร้าง Home.md
[ ] สร้าง Log.md
[ ] สร้าง _templates
```

อย่าเพิ่ง import ทุกอย่าง

อย่าเพิ่งติด plugin

Day 2: เอา source แรกเข้าระบบ

เลือก source จริงหนึ่งชิ้น

เช่น meeting note ล่าสุด

```
[ ] วาง source ใน 00-inbox
[ ] เพิ่ม date/source/context
[ ] ใช้ Source-to-Wiki Prompt
[ ] สร้าง Source note
[ ] อัปเดต Log
```

Day 3: สร้าง concept แรก

ดู source เมื่อวาน

ถามว่าเรื่องไหนใช้ซ้ำได้

```
[ ] เลือก concept หนึ่งเรื่อง
[ ] สร้าง Concept note
[ ] link กลับไป Source note
[ ] link ไป Project ถ้ามี
```

ตัวอย่าง:

```
AI training สำหรับคนไม่รู้โค้ด
```

Day 4: สร้าง project แรก

เลือกงานที่ต้องทำต่อจริง

```
[ ] สร้าง Project note
[ ] ใส่เป้าหมาย
[ ] ใส่ source ที่เกี่ยวข้อง
[ ] ใส่ todo 3-5 ข้อ
[ ] ใส่ deadline ถ้ามี
```

ตัวอย่าง:

```
FAQ หน้าเว็บคอร์ส AI
```

Day 5: ให้ Claude ช่วยร่างงานจาก Wiki

ส่งให้ Claude เฉพาะ note ที่เกี่ยวข้อง

```
[ ] Project note
[ ] Source note
[ ] Concept note
```

Prompt สั้น:

```
จาก note เหล่านี้ ช่วยร่าง output แรกสำหรับ project นี้
ใช้เฉพาะข้อมูลใน note
แยกส่วนที่เป็น fact กับส่วนที่เป็น draft/idea ให้ชัด
```

อย่าให้ Claude อ่านทั้ง vault

ให้ context เท่าที่จำเป็น

Day 6: ตรวจและกันพลาด

ใช้ checklist ก่อนส่งงาน

```
[ ] ตรวจ source
[ ] ตรวจข้อมูลที่ Claude แต่งเพิ่ม
[ ] แยก fact / inference / idea / decision
[ ] ลบข้อมูลลับที่ไม่จำเป็น
[ ] ปรับภาษาก่อนส่ง
```

ถ้าจะส่งงานจริง ให้คนที่เกี่ยวข้อง review ด้วย

Day 7: Weekly Review ครั้งแรก

ใช้ Weekly Review Checklist

```
[ ] ล้าง inbox
[ ] เช็ก project
[ ] หา note ซ้ำ
[ ] เช็ก link
[ ] อัปเดต Home
[ ] เขียน Log
```

จบ Day 7 คุณจะมี Wiki เล็ก ๆ ที่ใช้ได้จริงแล้ว

ไม่ต้องใหญ่

ไม่ต้องสวย

ขอให้กลับมาใช้ต่อได้

APPENDIX F

ชุดเริ่มต้นแบบสั้นที่สุด

ถ้าคุณอยาก copy เฉพาะสิ่งจำเป็นที่สุด ให้ใช้ชุดนี้

Folder

```
my-wiki
  00-inbox
  10-sources
  20-notes
  30-concepts
  40-projects
  90-index
  _templates
  Home.md
  Log.md
```

Note types

```
Source note   = ข้อมูลนี้มาจากไหน
Concept note  = เรื่องนี้คืออะไร
Project note  = งานนี้กำลังทำอะไร
Decision note = ตัดสินใจอะไรไปแล้ว
Index note    = เริ่มอ่านตรงไหน
```

Workflow

```
Capture → Clean → Ask Claude → Split → Link → Save → Review
```

บทบาท

> Obsidian = บ้านของความรู้
> Claude = ผู้ช่วยอ่านและจัดบ้าน
> คุณ = เจ้าของบ้านและ editor สุดท้าย

กติกากันพลาด

> 1. อย่าส่งข้อมูลลับถ้าไม่จำเป็น
> 2. อย่าเชื่อ Claude ถ้าไม่มี source
> 3. แยก Fact / Inference / Idea / Decision เสมอ
> 4. อย่าให้ระบบใหญ่กว่างาน
> 5. ทบทวนเป็นรอบ ไม่ใช่จัดครั้งเดียวแล้วจบ

ถ้าคุณจำได้แค่นี้ ก็เริ่มได้แล้ว

APPENDIX G

FAQ สั้นท้ายเล่ม

ต้องใช้ Obsidian เท่านั้นไหม

เล่มนี้ใช้ Obsidian เพราะเหมาะกับไฟล์ Markdown, link และ vault ที่คนคุมเองได้

แต่หลักคิด LLM Wiki ใช้กับเครื่องมืออื่นได้ ถ้าเครื่องมือนั้นทำ 4 อย่างนี้ได้:

> เก็บ note ได้
> link หรืออ้างอิง note ได้
> ค้นหาได้
> ส่ง context ให้ Claude อ่านได้

ถ้าคุณใช้เครื่องมืออื่นอยู่แล้ว ให้เริ่มจากหลักคิดก่อน ไม่ต้องย้ายทันที

ต้องใช้ Claude เท่านั้นไหม

เล่มนี้ใช้ Claude เป็นตัวอย่างหลัก

แต่ workflow ส่วนใหญ่ใช้กับ LLM อื่นได้

สิ่งสำคัญไม่ใช่ชื่อเครื่องมือ

สิ่งสำคัญคือคุณมี source, note, link, index และกติกาตรวจงานก่อนเชื่อ AI

ต้องทำทุก template ไหม

ไม่ต้อง

ถ้าเพิ่งเริ่ม ให้ใช้แค่ 3 อย่างก่อน:

```
Source note
Project note
Log.md
```

เมื่อเริ่มเจอเรื่องที่ใช้ซ้ำ ค่อยเพิ่ม Concept note

เมื่อมีการตัดสินใจสำคัญ ค่อยเพิ่ม Decision note

อย่าเริ่มจากการทำ template ครบแล้วไม่ได้ใช้

ควรทำ Wiki ส่วนตัวหรือ Wiki ทีม

เริ่มจากส่วนตัวก่อนง่ายกว่า

เพราะคุณคุมชื่อ note, folder และ workflow เองได้

เมื่อเห็นว่าใช้กับงานจริงได้ ค่อยชวนทีมใช้ project เดียว

อย่าเริ่มจากการบังคับทั้งทีมเปลี่ยนระบบพร้อมกัน

ถ้าไม่มีเวลาทำ Weekly Review ทำยังไง

ลดเหลือ 10 นาทีก็ได้

ทำแค่ 3 อย่าง:

```
1. ล้าง inbox ที่สำคัญที่สุด
2. เช็ก project active หนึ่งหน้า
3. เขียน Log สั้น ๆ ว่าอะไรค้าง
```

ทำเล็ก ๆ ดีกว่าไม่ทำเลย

ควรเก็บ chat กับ Claude ไหม

เก็บเฉพาะส่วนที่มีประโยชน์

ไม่ต้องเก็บทั้ง chat ทุกครั้ง

ให้ถามว่า:

> chat นี้มี prompt ที่ควรใช้ซ้ำไหม
> มี output ที่ตรวจแล้วและใช้ต่อได้ไหม
> มี concept หรือ decision ที่ควรเข้า Wiki ไหม

ถ้าไม่มี ก็ไม่ต้องเก็บ

ควรติด plugin เมื่อไหร่

เมื่อคุณรู้แล้วว่าปัญหาจริงคืออะไร

อย่าติด plugin เพราะคิดว่าจะทำให้ระบบดีเอง

ตัวอย่าง:

> ถ้าจัด template ซ้ำ ๆ บ่อย → ค่อยหา plugin/template workflow
> ถ้าเก็บเว็บบ่อย → ค่อยดู web clipper
> ถ้า search ไม่พอ → ค่อยปรับวิธีตั้งชื่อและ index ก่อน

เริ่มจาก workflow ก่อน plugin ทีหลัง

สัญญาณว่าเริ่มถูกทาง

คุณเริ่มถูกทางเมื่อ:

> หา note เดิมเจอเร็วขึ้น
> Claude ตอบตรงขึ้นเพราะมี context ดีขึ้น
> project note ทำให้กลับมาทำงานต่อได้ทันที
> source note ช่วยตรวจคำตอบได้
> Home ไม่ยาวเกินไป
> คุณยังใช้ระบบได้ในวันที่งานยุ่ง

ถ้าเกิดสิ่งเหล่านี้ แปลว่า Wiki เริ่มช่วยงานจริงแล้ว