

Framework: Value Per Hour (VPH)

วิธีวัด 100x ที่จับต้องได้จริงคือ **Value Per Hour**

$$\text{Value Per Hour} = \text{ผลลัพธ์ที่สร้างได้} \div \text{ชั่วโมงที่ใช้ไป}$$

 **ก่อน Automate**

Report 1 ฉบับ / 5 ชั่วโมง

0.2

Reports/ชั่วโมง

VPH เดิมที่ต่ำมาก

 **หลัง Automate**

52 Reports / 3 ชั่วโมง (setup)

86.5x

Productivity Gain

นั่นคือ 100x ในทางปฏิบัติ

ตารางเปรียบเทียบก่อน/หลัง Vibe Coding

ดูตัวเลขจริงที่เกิดขึ้นเมื่อนำ Vibe Coding มาใช้ในสถานการณ์ต่างๆ

สถานการณ์	ก่อน	หลัง	Multiplier
สร้าง Landing Page	จ้าง Dev 2 อาทิตย์ + 30,000 บาท	ทำเองใน 2 ชั่วโมง	80x (เวลา)
สร้าง Internal Tool	รอ IT 3 เดือน	สร้างเองใน 1 สัปดาห์	12x (เวลา)
ทำ Weekly Report	4 ชั่วโมง/สัปดาห์	0 ชั่วโมง (Automated)	∞
Test Business Idea	3 เดือน Build MVP	1 สัปดาห์	12x
แก้ Bug เล็กๆ	รอ Dev 3 วัน	แก้เองใน 30 นาที	14x



100x ไม่เกิดขึ้นพร้อมกัน — มันเป็น Compounding Effect

ผลลัพธ์ไม่ได้เกิดขึ้นในวันเดียว แต่สะสมทบต้นตามเวลา

1

เดือนที่ 1

ประหยัดเวลาได้ 2-3 ชั่วโมงต่อสัปดาห์

2

เดือนที่ 3

สร้าง Tool ที่ทำให้งานบางอย่างหายไปเลย

3

เดือนที่ 6

มี Side Project ที่สร้างรายได้เพิ่ม

4

ปีที่ 1

ทำสิ่งที่คนอื่นต้องใช้ทีม 5 คนได้คนเดียว — นั่นคือ 100x

Mindset ที่ต้องเปลี่ยนก่อนเริ่ม

"ฉันไม่ใช่คนเทคนิค ฉันทำไม่ได้หรอก"

นี่คือความคิดที่จะอุดรังคุณมากที่สุด เพราะ Vibe Coding ไม่ได้ถามว่าคุณ "เป็นคนเทคนิค" ไหม มันถามว่าคุณ:

1

รู้ไหมว่าต้องการอะไร?

คุณทำงานมาสิบปี คุณรู้ปัญหา
ของคุณดีกว่าใคร

2

สามารถอธิบายได้ชัดไหม?

ถ้าคุณอธิบายงานให้เพื่อนร่วม
งานได้ คุณก็อธิบายให้ AI ได้

3

ทดสอบและบอกได้ไหมว่า ผิดตรงไหน?

ทักษะนี้คือสิ่งที่คุณใช้ทุกวันในการ
ทำงานอยู่แล้ว

✔ ถ้าคุณทำงานมาสิบปีและรู้จักอุตสาหกรรมของคุณ — คุณมีทักษะสามข้อนี้ทั้งหมดแล้ว **คุณแค่ยังไม่รู้ว่าตัวเองมี**

Domain Knowledge คือทักษะที่สำคัญที่สุด

Vibe Coding ไม่ได้ต้องการให้คุณกลายเป็น Developer — มันต้องการให้คุณใช้สิ่งที่คุณมีอยู่แล้วให้เกิดประโยชน์สูงสุด



Domain Knowledge

ความรู้เชิงลึกในอุตสาหกรรมของคุณ
คือสิ่งที่ AI ไม่มี คุณรู้ว่าปัญหาคืออะไร
และผลลัพธ์ที่ดีหน้าตาเป็นอย่างไร



Problem Definition

การรู้ว่าปัญหาคืออะไรคือ 80% ของ
การแก้ปัญหา Vibe Coding ช่วยส่วน
ที่เหลืออีก 20%



Feedback & Iteration

ความสามารถในการบอกว่า "ผิดตรง
ไหน" คือทักษะที่มีค่ามากที่สุดในการ
ทำงานกับ AI

สรุป 3 บรรทัด

ทุกสิ่งที่คุณต้องจำจากบทนี้



100x ไม่ใช่ทำงานเร็วขึ้น 100 เท่า

มันคือขจัดงานไม่จำเป็น, ทำสิ่งที่
เป็นไปไม่ได้ให้ได้, และ Test Idea
ได้เร็วขึ้น



วัดด้วย Value Per Hour

เวลาที่ลงทุนในการ Automate จะ
คืนกลับมาเป็น 10-100 เท่าใน 1 ปี



**คุณมีทักษะที่สำคัญ
ที่สุดอยู่แล้ว**

Domain Knowledge และการรู้
ว่าปัญหาคืออะไร คือสิ่งที่ทรงพลัง
ที่สุด



พร้อมแล้วสำหรับบทถัดไป? เราจะเริ่มลงมือสร้างสิ่งแรกของคุณด้วย Vibe Coding

บทที่ 3 — Stack ที่ใช้สำหรับ Vibe Coder

คู่มือเลือกเครื่องมือ AI Coding ที่เหมาะกับคุณ — ตั้งแต่เริ่มต้นฟรีจนถึงระดับจริงจัง พร้อม Setup ใน 30 นาที



เรื่องราวที่เกิดขึ้นจริง

ปัญหาของ "ต้น"

ต้น อายุ 35 ปี เพิ่งเริ่ม Vibe Coding ใหม่ๆ เขาใช้เวลา 2 อาทิตย์แรก ลองเครื่องมือทุกอย่างที่มีบน Internet

- ChatGPT, Copilot, Cursor
- Replit, Bolt, v0, Lovable
- และอีกสิบตัว...

ผลที่ได้คือ...

งงมาก และยังไม่ได้สร้างอะไรเลย

ปัญหาไม่ได้อยู่ที่เครื่องมือ แต่อยู่ที่ ไม่มีกรอบในการเลือก

✔ บทนี้จะให้กรอบนั้นคุณ

หลักการเลือก Stack: ตามคำถามเดียว

ก่อนเลือกเครื่องมืออะไรก็ตาม ตามตัวเองว่า:

"ฉันต้องการ Output แบบไหน?"

เพราะเครื่องมือแต่ละตัวออกแบบมาเพื่อ Output ที่ต่างกัน



UI / Frontend

ต้องการสร้างหน้าเว็บสวยงาม หรือ Landing Page



Full App

ต้องการ App ที่มีทั้ง Frontend + Backend + Database



Personal Tool

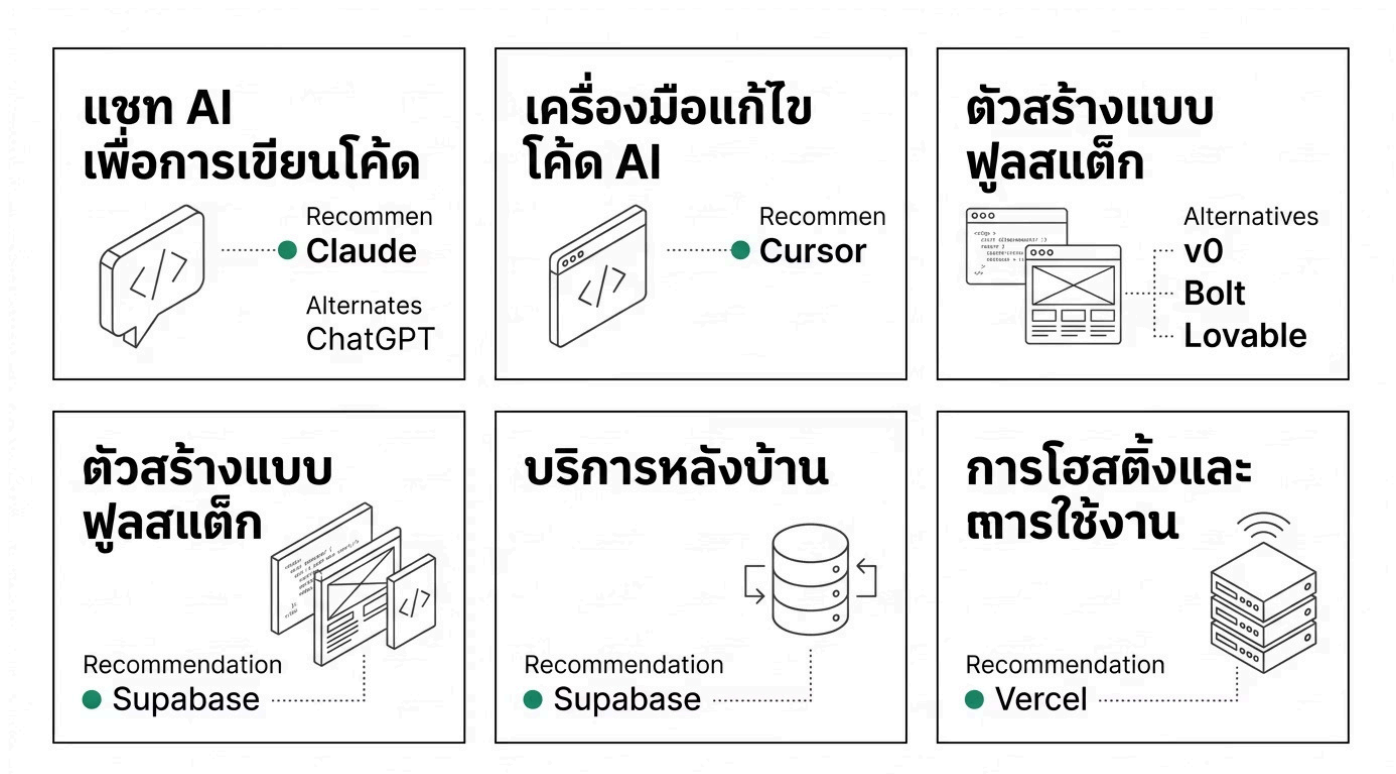
ต้องการ Script หรือ Tool ใช้งานส่วนตัว



Production App

ต้องการ App จริงจังที่ Deploy ให้คนอื่นใช้ได้

แผนที่เครื่องมือ Vibe Coding ปี 2025



เครื่องมือแต่ละหมวดมีจุดประสงค์ที่ต่างกัน การเลือกให้ถูกหมวดคือกุญแจสำคัญในการสร้าง Stack ที่มีประสิทธิภาพ

หมวดที่ 1 & 2: AI Chat + Code Editor

★ แนะนำสำหรับผู้เริ่มต้น

Claude (Anthropic)

- อธิบายละเอียด เข้าใจ Context ยาว ไม่ตัดทอน โค้ด
- เหมาะสำหรับ: วางแผนโปรเจกต์, เขียนโค้ดยาวๆ, อธิบายว่าโค้ดทำอะไร
- ราคา: ฟรี (มีจำกัด) / Pro \$20/เดือน

AI CHAT

ChatGPT (OpenAI)

- ใช้งานง่าย มีคนใช้เยอะจึงมี Tutorial เยอะ
- ราคา: ฟรี / Plus \$20/เดือน

★ แนะนำสำหรับโปรเจกต์จริงจัง

Cursor (AI Code Editor)

- แก้ไขหลายไฟล์พร้อมกัน เข้าใจ Codebase ทั้งโปรเจกต์
- ราคา: ฟรี (จำกัด) / Pro \$20/เดือน

หมวดที่ 3, 4 & 5: Builder + Backend + Hosting



★ UI/FRONTEND

v0 (Vercel)

สร้าง UI สวยมากจาก
Description ใช้ React +
Tailwind

v0.dev — ฟรี (จำกัด) /
\$20/เดือน



Bolt (StackBlitz)

สร้าง Full App ได้จาก
Prompt รับ Code ใน
บราวเซอร์ได้เลย

bolt.new — ฟรี (จำกัด)



★ แนะนำสุด

Supabase

Database + Auth +
Storage + API ในที่เดียว
Open Source

ฟรีถึง 500MB / Pro
\$25/เดือน



★ FRONTEND

Vercel

Deploy ด้วยคลิกเดียว
Fast CDN ทั่วโลก ฟรี
สำหรับโปรเจกต์เล็ก

Stack ที่แนะนำตามระดับและงบประมาณ

STACK A

เริ่มต้น (ฟรีเกือบทั้งหมด)

บทบาท	เครื่องมือ
AI Chat	Claude Free
Builder	Bolt.new
Database	Supabase Free
Hosting	Vercel Free
รวม	0 บาท/เดือน

STACK C

Minimal (Focused)

บทบาท	เครื่องมือ
AI Chat	Claude Pro
Editor	Cursor Pro
Database	Supabase Free
Hosting	Vercel Free
รวม	~1,440 บาท/เดือน

STACK B

จริงจัง (งบสมเหตุสมผล)

บทบาท	ราคา
Claude Pro	~720 บาท
Cursor Pro	~720 บาท
v0	~720 บาท
Supabase Pro	~900 บาท
Vercel Pro	~720 บาท
รวม	~3,800 บาท/เดือน



วิธีเลือกที่ถูกต้องสำหรับคุณ

1 คุณต้องการ Control มากแค่ไหน?

Control มาก → เลือก **Cursor + Claude** — แก้ไขโค้ดได้ทุกจุด เข้าใจทุก Layer

Speed มาก → เลือก **Bolt หรือ Lovable** — สร้าง App ได้ภายในชั่วโมง

2 งบประมาณต่อเดือนคือเท่าไร?

0 บาท → Stack A (Claude Free + Bolt + Supabase Free + Vercel Free)

1,000–2,000 บาท → Stack C (Claude Pro + Cursor)

3,000+ บาท → Stack B (Full Suite)

3 โปรเจกต์แรกของคุณคืออะไร?

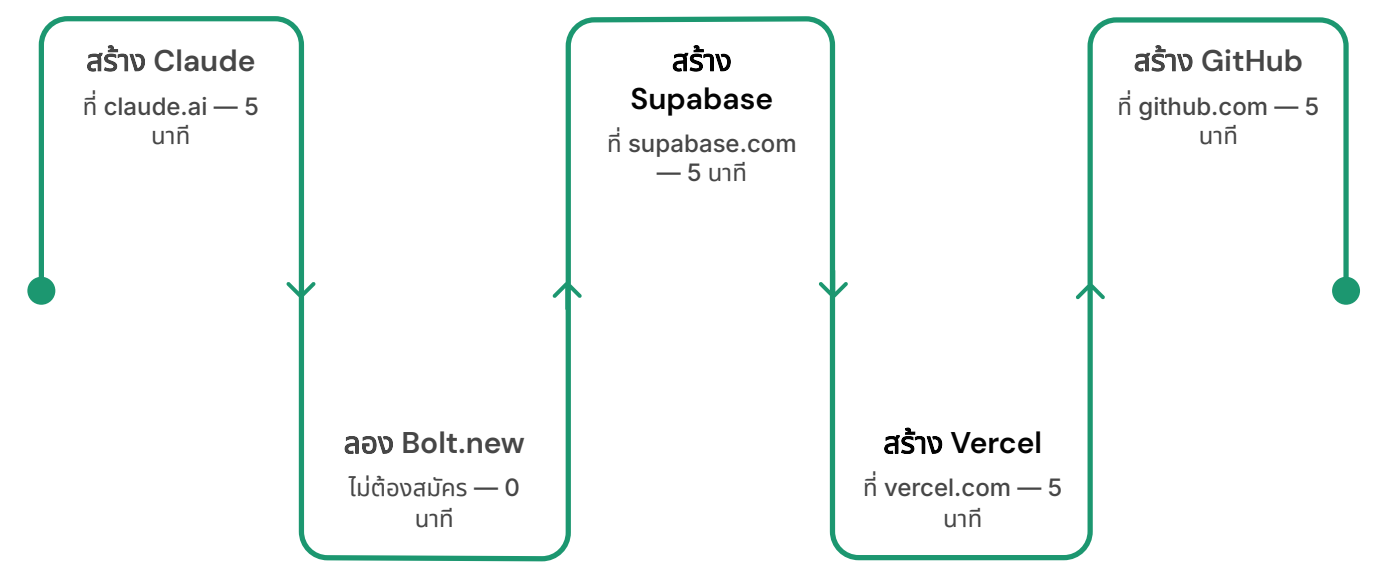
Landing Page / Static Site → **v0 + Vercel**

Web App พร้อม Database → **Bolt/Cursor + Supabase**

Tool ส่วนตัว → **Claude + Python Script**

Setup ใน 30 นาที (Stack A ฟรี)

ทำตามขั้นตอนนี้ได้เลยตอนนี้ — ไม่ต้องรอ ไม่ต้องเสียเงิน



ทั้งหมดนี้ใช้เวลาไม่เกิน 30 นาที และไม่มีค่าใช้จ่ายใดๆ คุณพร้อมเริ่ม Vibe Coding ได้ทันที

✓ Checklist การ Setup

- ☐ สร้าง Account Claude ที่ claude.ai
- ☐ ลอง Bolt.new (ไม่ต้อง Sign Up)
- ☐ สร้าง Account Supabase ที่ supabase.com
- ☐ สร้าง Account Vercel ที่ vercel.com
- ☐ สร้าง Account GitHub ที่ github.com

✓ เมื่อทำครบทั้ง 5 ขั้นตอน คุณมี Stack ที่พร้อมสร้าง Project จริงๆ ได้แล้ว โดยไม่เสียเงินแม้แต่บาทเดียว

สรุป 3 บรรทัด

1. เลือกตาม Output

เลือก Stack ตาม **Output** ที่ต้องการ ไม่ใช่ตาม Hype ว่าเครื่องมือไหนเป็นที่นิยมมากที่สุด

2. เริ่มต้นด้วยฟรีก่อน

เริ่มต้นด้วย **Stack ฟรี** ก่อนเสมอ แล้วค่อย Upgrade เมื่อคุณรู้ว่าต้องการอะไร

3. Combination ที่ใช้

Claude + Bolt + Supabase + Vercel คือ Combination ที่ทำ Project จริงๆ ได้โดยแทบไม่เสียเงิน



บทถัดไป: เราจะเริ่มสร้าง Project แรกด้วย Stack A ตั้งแต่ Prompt แรกจนถึง Deploy จริง — ไปต่อกันเลย



บทที่ 4 — คุยกับ AI ให้ได้งาน

Prompt ที่ใช้จริง สำหรับคนที่อยากให้ AI ทำงานแทนได้จริง ไม่ใช่แค่ตอบคำถาม

บทที่ 4

PROMPT ENGINEERING



เรื่องราวที่เกิดขึ้นจริง: พลอยกับ Claude

ครั้งแรก ❌

"พลอย" อายุ 33 ปี เปิด Claude ครั้งแรก พิมพ์ว่า
"ช่วยสร้างเว็บไซต์ให้หน่อย"

AI ตอบกลับมาเป็นคำถามสืบข้อ เธอหยุดหนิดและปิดแท็บ
ไป

สองอาทิตย์ต่อมา ✅

เธอลองใหม่ คราวนี้พิมพ์ว่า:

"ฉันเป็นเจ้าของร้านขายเสื้อผ้า Vintage ในกรุงเทพ
ต้องการหน้าเว็บสำหรับแสดงสินค้าใหม่ที่เพิ่งเข้ามา มี
รูปสินค้า ราคา และปุ่ม Line เพื่อสั่งซื้อ ใช้สีโทน Earth
Tone ดูอบอุ่น ไม่ต้องมีระบบ Cart ซับซ้อน แค่ให้คน
กด Line มาหาได้ สร้างด้วย HTML ที่ฉัน Upload ขึ้น
Netlify ได้เลย"

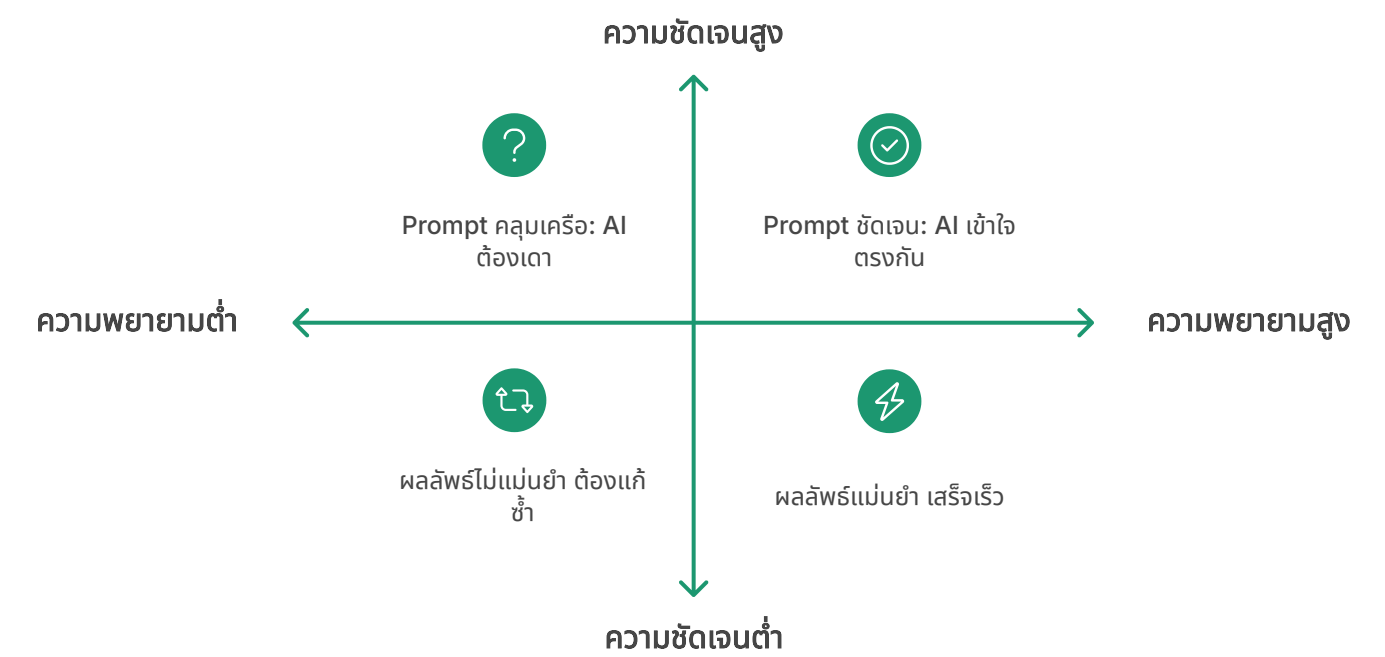
AI ส่ง Code กลับมาใน 30 วินาที เธอ Copy ไป Upload
— เว็บขึ้นจริงภายใน 10 นาที

✅ ความต่างคือ **คุณภาพของ Prompt** — ไม่ใช่ความฉลาดของ AI

ทำไม Prompt ถึงสำคัญขนาดนี้

AI ฉลาด แต่ AI ไม่ได้อ่านใจ ความชัดเจนของ Prompt คือสิ่งที่กำหนดคุณภาพของผลลัพธ์

Prompt คลุมเครือ	Prompt ชัดเจน
AI จะ "เดา" สิ่งที่คุณต้องการ	AI ทำงานแบบ "ช่างฝีมือ"
โอกาสถูกน้อยมาก ต้องแก้ไขหลายรอบ เสียเวลา	ผลลัพธ์แม่นยำมาก ได้งานตรงตามที่ต้องการทันที



การลงทุนเวลา 2 นาทีเพื่อเขียน Prompt ที่ดี ช่วยประหยัดเวลาแก้ไขได้หลายชั่วโมง

Framework: RCTF — สูตรสำเร็จของ Prompt ที่ได้ผล

RCTF คือโครงสร้าง 4 ส่วนที่ทำให้ AI เข้าใจสิ่งที่คุณต้องการได้อย่างแม่นยำ

 R — Role (บทบาท) บอก AI ว่าคุณต้องการให้มัน "เป็น" อะไร "คุณเป็น Senior Web Developer ที่เชี่ยวชาญ React และ Tailwind CSS" "คุณเป็น Python Developer ที่เขียนโค้ดสะอาด" "คุณเป็น UX Designer ที่เข้าใจผู้ใช้ทั่วไป"	 C — Context (บริบท) อธิบายสถานการณ์ ใครใช้ ใช้ทำอะไร มี Constraint อะไร "สร้างสำหรับร้านเสื้อผ้า Vintage ขนาดเล็กในกรุงเทพ" ลูกค้าส่วนใหญ่ใช้มือถือ ไม่ค่อยคุ้นเทคโนโลยี จบ Hosting จำกัด อยากรับบริการฟรีก่อน"
 T — Task (งาน) บอกสิ่งที่ต้องการอย่างชัดเจน ละเอียดย และเฉพาะเจาะจง ยิ่งละเอียดยิ่งดี	 F — Format (รูปแบบ Output) บอกว่าอยากได้ผลลัพธ์ในรูปแบบไหน "ส่งกลับมาเป็น Code Block เดียว ไม่ต้องอธิบายยาว" แต่ใส่ Comment สั้นๆ บอกว่าแต่ละส่วนทำอะไร"

Prompt ที่ไม่ได้ผล vs Prompt ที่ได้ผล

ดูตัวอย่างเปรียบเทียบ Prompt แบบคลุมเครือกับแบบที่ให้ผลลัพธ์จริง

✗ ไม่ได้ผล	✓ ได้ผล
"สร้างเว็บให้หน่อย"	"สร้าง Landing Page สำหรับ Yoga Studio ในกรุงเทพ มีหัวข้อ, ประโยชน์ 3 ข้อ, และ Form จอง Trial Class"
"แก้ Bug นี้"	"Function นี้ควรส่งคืน Array แต่ได้ undefined — นี่คือ Code และ Error Message"
"ทำให้สวยขึ้น"	"เพิ่ม Hover Effect บน Button, ทำ Card มีเงา, และเปลี่ยน Font เป็น Inter"
"ทำ Automation ให้"	"สร้าง Script Python ที่อ่านไฟล์ CSV ชื่อ orders.csv แล้วส่ง Email สรุปยอดขายวันไปที่ boss@company.com ทุกเที่ยงคืน"

📌 หลักการง่ายๆ: ยิ่งคุณบอกรายละเอียดมากเท่าไร AI ยิ่งทำงานได้ตรงใจมากเท่านั้น

Prompt Template ที่ใช้บ่อย – กลุ่ม UI / Frontend

Template 1: หน้าเว็บจาก Scratch

คุณเป็น Frontend Developer เชี่ยวชาญ HTML/CSS/Tailwind
ฉันต้องการ [ประเภทหน้าเว็บ] สำหรับ [ธุรกิจ/บริษัท]
กลุ่มเป้าหมายคือ [กลุ่มผู้ใช้]
หน้าเว็บต้องมี: [Feature 1], [Feature 2], [Feature 3]
สโลล: [บรรยายสโลล]
สร้างเป็น HTML ไฟล์เดียว Upload ขึ้น Netlify ได้เลย

Template 2: React Component

สร้าง React Component ชื่อ [ชื่อ Component]
หน้าที่: [อธิบายว่า Component ทำอะไร]
Props ที่รับ: [บอก Props]
Behavior: [อธิบาย Logic]
ใช้ Tailwind CSS

Template 4: ออกแบบ Database

ฉันกำลังสร้าง [ประเภทแอป] ที่ต้องการจัดการข้อมูลเกี่ยวกับ [ข้อมูลหลัก]
ช่วยออกแบบโครงสร้าง Database ที่เหมาะสมใช้ Supabase (PostgreSQL)
ส่ง SQL Script ที่รันบน Supabase ได้เลย

Template 7: Script อัตโนมัติ

สร้าง Python Script ที่:
1. [ขั้นตอนที่ 1]
2. [ขั้นตอนที่ 2]
Input: [ไฟล์/ข้อมูล]
Output: [ผลลัพธ์ที่ต้องการ]
อธิบาย Dependencies ที่ต้องติดตั้ง และวิธีรัน Script ด้วย

Template สำหรับ Debug และแก้ปัญหา

เมื่อ Code มีปัญหา การ Prompt ที่ถูกต้องช่วยให้ AI แก้ได้ตรงจุดและรวดเร็ว

Template 9: Debug Error Code นี้ได้รับ Error: [Error Message] Code ที่มีปัญหา: [วาง Code] Environment: [Language/Framework /Version] อธิบายว่า Error นี้เกิดจากอะไร และแก้ยังไง	สิ่งที่ต้องระบุเสมอ - Error Message ที่แน่นอน (Copy มาทั้งหมด) - Code ที่มีปัญหา (ไม่ใช่แค่บรรทัดเดียว) - Environment เช่น Python 3.11, Node 18, React 18 - สิ่งที่คาดหวังว่าควรเกิดขึ้น vs สิ่งที่เกิดขึ้นจริง	ผลลัพธ์ที่ได้ - AI อธิบายสาเหตุของ Error ได้ชัดเจน - ได้ Code ที่แก้ไขแล้วพร้อมใช้ - เข้าใจว่าทำไมถึงผิดและป้องกันในอนาคตได้
--	---	--

 เกล็ดลับ: ใส่ **Environment** ทุกครั้ง เพราะ Code ที่ถูกใน Python 2 อาจผิดใน Python 3

เทคนิคพิเศษ: "Vibe Language"

บางครั้งคุณไม่รู้ศัพท์เทคนิคที่ถูกต้อง — ไม่เป็นไร! AI เข้าใจภาษาคน สิ่งสำคัญคือบรรยายผลลัพธ์ที่ต้องการ ไม่ใช่วิธีการทำ



"อยากให้มันกระดิกๆ เวลา Hover"

AI เข้าใจว่าต้องการ **Animation Effect** เมื่อเมาส์ชี้ที่ Element



"ทำให้ดูหรูๆ แบบเว็บ Apple"

AI เข้าใจ **Aesthetic** ที่ต้องการ — สะอาด มีนิมอล มีพื้นที่ว่าง



"เวลาคลิกปุ่มแล้วขอให้มันรู้สึกลึกๆ ได้"

AI เข้าใจว่าต้องการ **Button Press Effect** — เช่น scale down หรือ shadow เปลี่ยน

หลักการ Vibe Language: บรรยายความรู้สึกหรือผลลัพธ์ที่ต้องการ แทนที่จะพยายามใช้ศัพท์เทคนิคที่ไม่แน่ใจ AI จะแปลงภาษาคนเป็นโค้ดให้เอง

เมื่อ AI งง หรือตอบไม่ตรง — 5 วิธีแก้



Break ลงอีก

ถ้า AI ทำงานชิ้นใหญ่ไม่ได้ → แบ่งเป็น
ชิ้นเล็กกว่า ทำทีละขั้นตอน



ให้ตัวอย่าง

หาตัวอย่างที่คล้ายกับที่ต้องการแล้ว
บอกว่า "อยากได้แบบนี้" — AI จะเข้าใจ
ทันที



บอกว่าผิดตรงไหน

"ส่วนของ X ผิด เพราะ Y ควรเป็น Z"
— ระบุจุดที่ผิดให้ชัดเจน ไม่ต้องเริ่มใหม่
ทั้งหมด



เริ่มใหม่

เปิด Chat ใหม่แล้ว Prompt ใหม่ที่ชัดกว่า บางครั้ง
Context เก่าทำให้ AI สับสน



ถามก่อน Build

บอกว่า "ก่อนเริ่มเขียน Code บอกฉันก่อนว่าคุณเข้าใจว่า
ต้องทำอะไร" — ป้องกันความเข้าใจผิดตั้งแต่ต้น