AI CODING ROADMAP • 2026

دليلك العملي لاستخدام الـAI في البرمجة

مش المطلوب إنك تجرب كل Tool. المطلوب إنك تتعلم أداة واحدة صح، تستخدمها على مشروع حقيقي، وبعدها تنتقل للمرحلة اللي بعدها.

OpenCode → Grok → Claude Code → Codex

روابط فيديوهات مرتبة + شرح كل مرحلة + نصائح تطبيق عملية

Prepared for Ahmed Alaydi

ابدأ بالطريقة الصح

أكبر غلطة في الـAI Coding إنك تنزل 5 أدوات في نفس اليوم، وتشوف ساعتين Tutorial لكل واحدة، وفي الآخر ما تدخلش أي أداة في شغلك الحقيقي.

الترتيب المقترح

1. OpenCode	افهم فكرة الـCoding Agent والـTerminal workflow
2. Grok	استخدم AI كمساعد في Debugging وشرح وتحسين الكود
3. Claude Code	ادخل على Project حقيقي ثم Skills + MCP
4. Codex	تعلم الـCLI والـAgents والـPrompts والـAGENTS.md

> **قاعدة مهمة:** بعد كل فيديو، افتح Project صغير وطبّق نقطة واحدة على الأقل. المشاهدة وحدها مش Skill.

إيه اللي لازم تكون فاهمه قبل أي Agent؟

- تعرف تستخدم Git وGitHub بشكل أساسي.
- تفهم الـProject structure بتاعك بدل ما تسيب الـAI يقرر كل حاجة.
- تعرف تشغل Tests أو على الأقل تختبر الـFeature بنفسك.
- ما تشاركش Secrets أو API Keys داخل Prompt عشوائي.

01 OpenCode - البداية مع الـCoding Agents

لو عايز تبدأ بأداة Open Source وتفهم فكرة إن الـAgent يقرأ المشروع، يخطط، يعدّل ملفات، ويستخدم Skills وMCP، فده اختيار مناسب كبداية.

الفيديو الأساسي

Leon van - **OpenCode Tutorial for Beginners: Setup, Agents, Skills & MCP** Zyl

يغطي التثبيت، الـSubagents، AGENTS.md، Skills، Free Models وMCP مع مثال عملي.

https://www.youtube.com/watch?v=uZGDO0L-Dr4

Bonus - تحديث OpenCode 2.0

لو خلصت الأساسيات، شوف التحديث الأحدث عشان تعرف الـDesktop sessions والـBackground agents والـLocal models.

https://www.youtube.com/watch?v=OyuId4YQiIs

طبّق بعد الفيديو: افتح Project صغير، اطلب من OpenCode يعمل Plan فقط بدون تعديل، راجع الخطة، وبعدها خليه ينفذ Feature واحدة.

02 Grok - استخدمه كمساعد برمجي

Grok مناسب كمرحلة لفهم استخدام الـAI في توليد كود، Debugging، شرح Errors، Refactoring وكتابة Tests. الفكرة هنا إنك تتعلم **إزاي تسأل صح** قبل ما تنتقل لـAgent يتحكم في المشروع.

الفيديو المقترح

How To Use Grok AI For Coding [Step-By-Step]

مناسب للمبتدئ: refactoring، explanations، debugging، generating code، tests، performance وdocumentation.

https://www.youtube.com/watch?v=R7tRI85j6mU

Prompt عملي بعد الفيديو

```
I am working on a [framework/language] project.
Do not rewrite the whole solution immediately.
First explain the bug, identify the likely root cause, then propose the smallest safe
fix.
After that, show me how to verify the fix with a test.
```

متستخدمش Grok أو أي Chat AI كـCopy/Paste machine. اسأله يشرح ليه الحل صح، وإيه السيناريوهات اللي ممكن تكسره.

03 Claude Code - المرحلة 1: البداية

هنا تبدأ تتعامل مع Agent جوه بيئة التطوير بدل Chat منفصل. الهدف إنك تفهم الـpermissions، الـcontext، طريقة إعطاء Tasks، ومتى تخليه يخطط قبل التنفيذ.

الفيديو 1 - Beginner

Tech With Tim - **FULL Claude Code Tutorial for Beginners in 2026**

https://www.youtube.com/watch?v=qYqIhX9hTQk

Bonus - Claude 101 عربي

كورس تعلم كلود من الصفر | Claude 101 in Arabic

اللينك اللي أرسلته - استخدمه كمرجع عربي إضافي لو عايز تثبت المفاهيم قبل الدخول للـworkflow المتقدم.

https://youtu.be/oz44Wp22iCU?si=ycVVFFXsJje5_iR8

هدف المرحلة: خليه يقرأ المشروع ويشرح الـarchitecture قبل ما تطلب أي تعديل كبير.

04 المرحلة 2: Project حقيقي - Claude Code

بعد ما فهمت الأساسيات، لازم تشوف Workflow حقيقي: Git، Refactoring، Tests، Feature وCI/CD. هنا يبدأ الفرق بين استخدام AI كـChat وبين استخدامه كـEngineering Agent.

الفيديو 2 - Build Apps 10x Faster

Programming with - **Claude Code Tutorial - Build Apps 10x Faster with AI** Mosh

مثال Full Stack عملي مع Refactoring، Testing، Roles، Authentication وGitHub Actions.

https://www.youtube.com/watch?v=IuyVVtr1uhY

طبّق بعد الفيديو: اختار Feature موجودة أصلًا عندك واطلب: Plan → Audit → تنفيذ صغير → Tests → Review. ما تبدأش Redesign أو Refactor كامل من أول Task.

05 المرحلة 3: Skills + MCP - Claude Code

دي المرحلة اللي تخلي Claude Code أكتر من Agent بيكتب Skills. Code بتضيف Workflow وتعليمات متخصصة، وMCP بيوصل الـAgent بأدوات ومصادر خارجية.

الفيديو 3 - Advanced

Tech With Tim - **The Ultimate Claude Code Guide | MCP, Skills & More**

https://www.youtube.com/watch?v=uogzSxOw4LU

بعد الفيديو جرّب

- Skill واحدة فقط مرتبطة بشغلك الحالي.
- MCP واحد له قيمة واضحة، مثل Browser testing أو Docs.
- ملف قواعد للمشروع بدل ما تعيد نفس التعليمات كل مرة.

ما تركبش 20 Skill و10 MCP Servers مرة واحدة. كل إضافة بتزود قدرات، لكن ممكن تزود Noise وصلاحيات ومشاكل Debugging.

06 Codex - المرحلة 1: الأساسيات

في Codex، ركز الأول على فهم طريقة إعطاء الـTask، قراءة الـrepo، الـapprovals، وتشغيل الـcommands بشكل آمن.

الفيديو 1 - Beginner Guide

OpenAI Codex Complete Beginner Guide

https://www.youtube.com/watch?v=pMOMqZJYJ5c

هدف المرحلة: Task صغيرة لها Definition of Done واضح. مثال: “Fix validation bug + add regression test + do not change API contract”.

07 Codex - المرحلة 2: مشروع عملي

أفضل طريقة تتعلم Agent هي تشوفه في Task حقيقية. الفيديو الرسمي ده بيعرض استخدام Codex CLI في بناء مشروع وتنفيذ Plan لحد التشغيل.

الفيديو 2 - OpenAI Official

Using OpenAI Codex CLI with GPT-5-Codex

فيديو عملي من OpenAI: Planning، CLI commands، Implementation ثم Deploy.

https://www.youtube.com/watch?v=iqNzfK4_meQ

اتفرج بعقلية Workflow: إزاي اتحدد المطلوب؟ إزاي الـAgent خطط؟ فين الإنسان راجع؟ وإزاي تم اختبار النتيجة؟

08 المرحلة 3: Codex + Prompts + Skills + AGENTS.md

بعد الأساسيات والمشروع الحقيقي، ابدأ في تحسين التعليمات نفسها. الـAgent القوي مع Instructions ضعيفة ممكن يطلع نتيجة أسوأ من Agent أبسط مع Context مضبوط.

الفيديو 3 - Advanced Workflow

GPT-6 Astra in Codex: 7 Ways to Improve Prompts, Skills, and AGENTS.md

https://www.youtube.com/watch?v=FESNAh3gNyQ

ركّز على 4 حاجات

- عرّف معنى "خلصت" بوضوح.
- حدد الحاجات اللي محتاجة موافقتك.
- اربط الـTask بالـdocs والـtests المطلوبة.
- خلي كل Skill لها Trigger واضح بدل تعليمات عامة.

مصادر عربية إضافية أرسلتها

المصادر دي مش جزء إجباري من المسار، لكنها مفيدة لو عايز شرح عربي إضافي أو زاوية مختلفة قبل/بعد الفيديوهات الإنجليزية.

1. Claude 101 in Arabic

https://youtu.be/oz44Wp22iCU?si=ycVVFFXsJje5_iR8

2. انسى Claude

احتفظت بالعنوان كما يظهر على YouTube. استخدمه كمحتوى إضافي، وتأكد من موضوعه الحالي من الوصف قبل ما تحطه في مرحلة محددة من الـRoadmap.

https://youtu.be/NXvqxP1bjf4?si=87gNgUddDAeZm6x3

> لو هتعمل Screenshot للفيديوهات، خليه يوضح اسم الـTool والعنوان بوضوح، ومتخليش اللقطة تعتمد على Thumbnail فيها Claim مبالغ فيه.

إزاي تستخدم الـAI في شغلك من غير ما يبوظ مستواك؟

1. افهم المشكلة

قبل الـPrompt، اكتب المشكلة في سطرين بإيدك.

2. ادّي Context كفاية

Framework، الملفات المهمة، القيود، والسلوك المتوقع.

3. Plan قبل التنفيذ

خصوصًا في الـrefactor والـarchitecture والتعديلات الكبيرة.

4. Task صغيرة

قلل مساحة الخطأ وخلي الـReview أسهل.

5. Tests

اطلب إثبات للنتيجة، مش مجرد "Done".

6. راجع الـDiff

متقبلش 30 ملف متغير من غير سبب واضح.

Prompt Template عملي لأي Coding Agent

```
Context:
- Project: [stack / framework]
- Goal: [what should change]
- Constraints: [what must NOT change]

Before editing:
1. Inspect the relevant files.
2. Explain the root cause / current architecture.
3. Propose a minimal plan.

Then implement only after the plan is clear.
Run the relevant tests and summarize the exact files changed.
```

خطة تعلم سريعة - 7 أيام

اليوم 1: OpenCode Tutorial + تثبيت وتجربة Task صغيرة.

اليوم 2: Grok Coding + Debugging لخطأ حقيقي عندك.

اليوم 3: Claude Code Beginner + قراءة Project موجود.

اليوم 4: Claude Code Project Workflow + Feature واحدة كاملة.

اليوم 5: Claude Skills/MCP + إضافة واحدة فقط.

اليوم 6: Codex Beginner + الفيديو العملي من OpenAI.

اليوم 7: Codex Advanced Prompting + كتابة AGENTS.md / Project instructions.

> **في النهاية:** اختار Agent أساسي واحد للشغل اليومي. الباقي خليه أدوات تعرف تستخدمها وقت الحاجة، مش حاجة لازم تفتحها كلها كل يوم.

FINAL NOTE

الأداة مش هي اللي هتخليك Developer أقوى

القيمة الحقيقية إنك تعرف تحلل المشكلة، تدي Context صح، تراجع القرار، وتتحمل مسؤولية النتيجة. الـAI يسرّعك، لكنه ما يعوضش فهمك.

تعلم → طبّق → راجع → حسّن → كرر

Ahmed Alaydi