

שנה ל-Claude Code: Boris Cherny על פיתוח בלי לכתוב קוד

שיחת פודקאסט עם יוצר Claude Code: מה השתנה בשנה האחרונה, איך הצוות שלו עובד, ולמה כדאי לבנות מוצר למודל של עוד חצי שנה

CyrilXBT על ידי X · Boris Cherny, Lenny Rachitsky · Lenny's Podcast (לפי התמליל; הועלה מחדש ב-X על ידי CyrilXBT)
· הועלה 2026-05-22

משך: כ-83 דקות · X

סיכום אוטומטי — (aibriefing.dev) AI Briefing

על המסמך הזה

זו שיחת פודקאסט ארוכה בין מגיש שמוצג בתמליל רק בשם Lenny (ככל הנראה Lenny Rachitsky, כלומר Lenny's Podcast; שם המשפחה לא נאמר) לבין Boris, האיש שהקים את Claude ב-Anthropic. לפי המגיש, הפרק יוצא בערך במלאת שנה להשקת Claude Code. זה ראיון ולא הרצאה, ולכן אין שקפים. על המסך מופיעים רק שני הדוברים, כמה כותרות עיתונות ופרסומות של Metaview. הסרטון הגיע מהעלאת מחדש ב-X, והכיתוב שצירף מי שהעלה אותו (למשל ש-CLAUDE.md גוזל 14%) לא מופיע בתמליל בכלל. לכן לא השתמשנו בו.

הטענה המרכזית של Boris: כתיבת קוד היא בעיה שכמעט נפתרה. הוא עצמו לא ערך שורת קוד ידנית מאז נובמבר. מה שמבדיל עכשיו צוותים ומוצרים זה איך עובדים מול מודל שמשתפר בקצב אקספוננציאלי: לתקצב צוותים קצת מתחת למה שצריך, לזוז מהר, לחפש latent demand אצל המשתמשים וגם אצל המודל, לא לכלוא את המודל ב-workflows נוקשים, ולבנות למודל של עוד חצי שנה. ברקע עובר גם הנרטיב של Anthropic: קודם coding, אחר כך tool use, אחר כך computer use, והבטיחות כמנוע של כל הרצף.



שני הדוברים בפורמט מסך מפוצל: המגיש משמאל, Boris (בקפוצ'ון) מימין. זה פורמט כל הפרק, ואין בו שקפים

איך לקרוא את המסמך

- 1. **תקציר מנהלים:** הטענות המרכזיות עם חותמות זמן, ומה שלא נאמר.
- 2. **החזרה ל-Anthropic והמספרים:** למה Boris חזר מ-Cursor ומה המשמעות של נתון ה-4% מה-commits.
- 3. **מהאק בטרמינל למוצר:** איך Claude Code נולד, למה דווקא בטרמינל, והתחזית על ה-IDE.
- 4. **העבודה היומיומית:** 10 עד 30 PRs ביום, code review אוטומטי, ו-Claude שמציע מה לבנות.
- 5. **המקצוע משתנה:** מכונת הדפוס כאנלוגיה, שחיקת מיומנויות, ולמה כדאי להיות generalist.
- 6. **Latent demand, Cowork ובטיחות:** העיקרון שהוא מגדיר כחשוב ביותר במוצר, Cowork שנבנה בעשרה ימים, ושכבות הבטיחות.
- 7. **עקרונות לבניית מוצרי AI וטיפים ל-Claude Code:** bitter lesson, plan mode, ואיזה מודל כדאי להפעיל.
- 8. **מה עושים עם זה, ו-9. מילון מונחים.**

הערת מקור המסמך הופק אוטומטית מהתמליל ומהפריימים של ההקלטה. התמליל עבר ASR, ולכן שמות ומונחים עלולים להופיע משובשים (למשל quad code או Cloud Code במקום Claude Code, ו-Strauss במקום Stross). הציטוטים מועתקים מילה במילה מהתמליל, כולל שיבושי ה-ASR, ונבדקו מול חותמת הזמן שלהם. בתמליל יש קטעים חסרים של כדקה וחצי, ולכן חלק מהטיעונים קטועים. שום דבר לא נוסף ממקורות חיצוניים.

1. תקציר מנהלים

Boris מתאר שנה שבה Claude Code עבר מהאק פרטי בטרמינל לכלי שכותב, לפי 4% SemiAnalysis, מה-commits הציבוריים ב-GitHub. רוב השיחה לא עוסקת במספרים. היא עוסקת בשיטת העבודה שמאחוריהם: מה הצוות שלו עושה אחרת, ואיך כדאי לבנות מוצר על מודל שמשתנה כל כמה חודשים.

- **Claude כותב את כל הקוד, אבל בני אדם עדיין קוראים אותו:** Boris מעלה 10 עד 30 PRs ביום, כולם נכתבו ב-Claude Code, ולא ערך שורה ידנית מאז נובמבר. ב-Anthropic, Claude עושה review ל-100% מה-PRs, ויש אחריו עוד שכבת review אנושית [12:22].
- **הצמיחה מאיצה:** 4% מה-commits הציבוריים, ולדבריו בריפוזיטורז פרטיים השיעור גבוה יותר. קצב הצמיחה ממשיך לעלות בכל מדד [1:31].
- **לתקצב מתחת למה שצריך זה יתרון:** כשמהנדס אחד לבד על פרויקט, הוא נאלץ להעביר עבודה ל-Claude. Claude Code עצמו נבנה בטרמינל כי בהתחלה Boris עבד עליו לבד [6:12], [20:03].
- **latent demand הוא לדבריו העיקרון החשוב ביותר במוצר:** Cowork נולד מאנשים שהשתמשו ב-Claude Code למשימות לא טכניות. בגרסה החדשה של העיקרון מסתכלים גם על מה שהמודל מנסה לעשות, ולא רק על מה שהשתמשים עושים [41:39], [46:13].
- **Cowork נבנה בכעשרה ימים, כולו עם Claude Code:** כולל מערכת guardrails ו-VM שלם שנשלח עם המוצר. בניגוד ל-Claude Code, הוא תפס מיד [47:47].
- **scaffolding מחזיק מעמד עד המודל הבא:** scaffolding workflows ו-scaffolding משפרים ביצועים בכ-10 עד 20%, והשיפור נמחק בדור המודל הבא. לכן הוא מציע לבנות למודל של עוד שישה חודשים [1:00:10].
- **המודל החזק ביותר הוא לרוב גם הזול ביותר:** Opus 4.6 על maximum effort צורך פחות טוקנים לאותה משימה, כי הוא צריך פחות תיקונים. כ-80% מהמשימות שלו מתחילות ב-plan mode [1:04:50].
- **משך ריצה אוטונומית קפץ פי כמה:** Sonnet 3.5 החזיק 15 עד 30 שניות לפני שיצא מהפסים. Opus 4.6 רץ 10 עד 30 דקות בלי השגחה, ולפעמים שעות או ימים [1:03:19].
- **coding נפתר, והחזית עוברת לעבודה שאינה קוד:** Cowork משלם לו דוחות, מנהל את הפרויקטים של הצוות ושולח תזכורות ב-[1:17:07], [13:53] Slack.

מה לא נאמר

- **CLAUDE.md ו-14%:** הכיתוב של ההעלאה ב-X מבטיח הסבר על 14% שהולכים ל-CLAUDE.md. בתמליל לא מוזכרים CLAUDE.md, context window או המספר 14%.
- **אין נתונים על איכות:** לא נמסרו שיעורי באגים, revert או תקלות בקוד שנכתב על ידי Claude. אין גם הסבר איך ה-review האוטומטי בנוי.
- **מספרי ההכנסות באים מהמגיש:** 2 מיליארד דולר ל-Claude Code ו-15 מיליארד ל-Anthropic נאמרו על ידי המגיש בניסוח של "Boris. I think I לא אישר אותם [1:11:01]."
- **אין מספרי עלות:** הוא מזכיר מהנדסים שמוציאים מאות אלפי דולרים בחודש על טוקנים, אבל לא אומר כמה אנשים כאלה יש ומה התשואה [23:12].
- **שכבת הבטיחות השלישית קטועה:** התמליל נקטע אחרי השכבה השנייה (evals). את השלישית, שחרור מוקדם, אנחנו מכירים רק מהסיכום של המגיש [52:26].
- **4% ו-20%:** מקור הנתונים הוא SemiAnalysis, והם לא נבדקו בשיחה. גם התחזית ל-100% קוד שנכתב על ידי AI מבוססת על הארכת קו מגמה, והוא אומר את זה במפורש [9:19].

2. החזרה ל-Anthropic והמספרים

השיחה נפתחת בתקופה הקצרה של Boris ב-Cursor. הוא משבח את הצוות ואת המוצר, אבל אומר שגילה מהר שמה שחסר לו הוא המשימה של Anthropic, ובמיוחד הבטיחות. לדבריו, כל מי ששואלים במסדרון ב-Anthropic למה הוא שם יענה [0:00] safety.

אחר כך המגיש מציג את הדוח של SemiAnalysis (על המסך מופיעה הכותרת Claude Code is the Inflection Point): 4% מה-commits הציבוריים ב-GitHub נכתבים היום על ידי Claude Code, והתחזית היא חמישית מהם

עד סוף השנה. לצדו מוצגת כותרת של Spotify, שלפיה המפתחים הטובים שלה לא כתבו שורת קוד מאז דצמבר. Boris מוסיף שתי הסתייגויות לטובתו: הנתון מתייחס רק לריפוזיטוריו ציבוריים, וקצב הצמיחה עצמו מאיץ [1:31].

Boris Cherny [1:31] These are also just public commits. So we actually think if you look at private — repositories, it's quite a bit higher than that

3. מהאק בטרמינל למוצר

ב-Anthropic, לדברי Boris, המודלים נבנים לאורך מסלול קבוע: קודם coding, אחר כך tool use, ואחר כך computer use. הצוות שבו הוא התחיל, Anthropic Labs, בנה את Claude Code, את MCP ואת אפליקציית הדסקטופ. לדבריו Mike Krieger ו-Ben Mann מפעילים עכשיו את הצוות הזה מחדש [3:06]. הוא בילה חודש בבניית אבות טיפוס, ורובם לא יצאו לאוויר.

הטרמינל לא היה החלטה עיצובית. הוא היה הדרך הקלה לבנות כשאתה לבד, ואחר כך נשארו בו כי לא הייתה תצורה אחרת שהצליחה לעמוד בקצב השיפור של המודל. Ben Mann דחף אותו לבנות גרף DAU עוד לפני ההשקה הפנימית, והגרף זינק מיד. ההשקה החיצונית בפברואר לא הייתה להיט: לקח חודשים עד שאנשים הבינו מה המוצר הזה עושה [6:12]. היום Claude Code זמין ב-iOS, ב-Android, בדסקטופ, בוובר, כתוסף ל-IDE, ב-Slack וב-GitHub.

בכנס Code with Claude במאי 2025 הוא חזה שעד סוף השנה לא יהיה צורך ב-IDE, והקהל, לדבריו, נשם בתדהמה. ההסבר שלו לתחזית: Anthropic חושבת באקספוננטים, ושלושה מהמייסדים היו שלושת המחברים הראשונים של מאמר ה-scaling laws. הוא רק המשיך את קו המגמה של אחוז הקוד שנכתב על ידי Claude. אצלו זה קרה בנובמבר [9:19].

Boris Cherny [6:12] It's like, you want to under-resource things a little bit at the start —

4. העבודה היומיומית: 100% קוד מ-Claude, ומה הלאה

התיאור הכי קונקרטי בשיחה הוא שגרת העבודה של Boris: 10 עד 30 PRs ביום, כולם נכתבו ב-Claude Code. הוא עדיין קורא את הקוד, כי לדבריו עדיין צריך לוודא שהקוד נכון ובטוח, חוץ מקוד פרוטוטיפי שלא ירוץ בשום מקום [12:22].

לדבריו, הצעד הבא כבר מתרחש: Claude עובר על פידבק, על דיווחי באגים ועל טלמטריה, ומציע בעצמו תיקונים ופיצ'רים. בשיטה שלו מפנים את Claude Code או את Cwork לערוץ Slack של הפידבק הפנימי, ו-Claude חוזר עם כמה PRs מוכנים. בימים הראשונים Boris תיקן כל פידבק תוך דקות, כדי שאנשים ירגישו שמקשיבים להם וימשיכו לשלוח [15:27].

הסיפור על דליפת הזיכרון ממחיש את השינוי בהרגלים. Boris עצמו ניתח heap snapshot בכלים ייעודיים. מהנדס חדש בצוות פשוט ביקש מ-Claude Code לטפל בזה. Claude לקח snapshot, כתב לעצמו כלי ניתוח just-in-time ופתח PR מהר יותר מ-Boris. המסקנה שלו היא שמי שעובד עם המודלים הרבה זמן צריך להזכיר לעצמו שזה כבר לא [18:31] Sonnet 3.5. משם עוברים לעקרונות של הצוות: לתקצב מתחת למה שצריך, לזוז מהר (מה שאפשר לעשות היום עושים היום), ולשאול קודם אם Claude יכול לעשות את זה [20:03]. אופטימיזציה של עלות טוקנים עושים רק כשמשהו כבר עובד ויקר. לדבריו, יש ב-Anthropic מהנדסים שמוציאים מאות אלפים בחודש על טוקנים [23:12].

Boris Cherny [13:53] I think at this point, it's safe to say that coding is largely solved, at least for — the kinds of programming that I do is just a solved problem because Quad can do it

5. המקצוע משתנה: מכונת הדפוס, שחיקה ו-generalists

Boris מספר שלמד לתכנת כדי לרמות במבחן במתמטיקה עם מחשבון TI-83 Plus. הוא כתב ספר על TypeScript, הקים meetup של TypeScript שבזמנו היה לדבריו הגדול בעולם, ולא למד מדעי המחשב באופן פורמלי. בשבילו קוד תמיד היה כלי ולא מטרה. יחד עם זה הוא מזכיר את Lena מהצוות, שעדיין כותבת ++C ידנית בסופי שבוע מתוך

הנאה [23:12], [24:42]. שחיקת מיומנויות לא מדאיגה אותו. תכנות תמיד עבר שלבים: עט ונייר, מתגים, punch cards ותוכנה. לדבריו עוד כשנה כדאי להבין את השכבה שמתחת, ואחר כך זה יהיה כמו [26:14] assembly.

האנלוגיה המרכזית שלו היא מכונת הדפוס. עלות החומר המודפס ירדה בערך פי 100 במשך 50 שנה, והאוריינות הגיעה ל-70% בעולם במשך 200 שנה. הוא מביא ראיון עם סופר מהמאה ה-15 ששמח שלא יצטרך עוד להעתיק ספרים ויוכל להתמקד באיור ובכריכה, וככה הוא מרגיש לגבי קוד [29:18]. העצה שלו: להתנסות בכלים בלי פחד, ולהיות generalist. בצוות Claude Code כולם כותבים קוד, כולל ה-PM, מנהל ההנדסה, המעצב, איש הכספים וה-data scientist. לדבריו כבר יש חפיפה של כ-50% בין התפקידים [37:03]. המגיש מוסיף סקר לא רשמי שערך ב-Twitter: 70% מהמהנדסים ומה-PMs נהנים מהעבודה יותר מאז שהתחילו להשתמש בכלי AI, ובערך 10% נהנים פחות. אצל מעצבים התוצאה הייתה 55% מול 20% [40:06].

Boris Cherny [37:03] So on the cloud code team, everyone codes, you know, our product — manager codes, our engineering manager codes, our designer codes, our finance guy codes, our data scientist codes, like everyone on the team codes

6. Latent demand, Cowork ושכבות הבטיחות

Boris מגדיר latent demand כמצב שבו אנשים משתמשים במוצר לדבר שהוא לא תוכנן בשבילו, וזה מאותת לא המוצר צריך להתפתח. הדוגמאות שלו מגיעות מ-Facebook: Marketplace נולד מהתצפית שבערך ב-2016 כ-40% מהפוסטים בקבוצות היו קנייה ומכירה, ו-Facebook Dating נולד מהתצפית שכ-60% מצפיות הפרופיל היו בין אנשים שאינם חברים [43:09]. בצוות שלו זה קרה כש-data scientists התחילו להריץ ניתוחים בטרמינל דרך Claude Code [46:13].

החידוש שלו הוא latent demand כלפי המודל: להסתכל על מה שהמודל מנסה לעשות ולהקל עליו. במחקר קוראים לזה being on distribution. ב-Claude Code הם הפכו את הגישה המקובלת: במקום להכניס את המודל כרכיב בתוך אפליקציה, המודל הוא המוצר, וסביבו יש מינימום scaffolding ומינימום כלים [46:13].

Cowork הוא התוצר של העיקרון הזה. הצוות ניסה כמה חודשים כיוונים שונים, ובסוף מישהו הציע פשוט לשים את Claude Code בתוך אפליקציית הדסקטופ. הבנייה לקחה כעשרה ימים, כולה עם Claude Code, כולל מערכת guardrails ו-VM שלם שנשלח עם המוצר. הוא מייחס את הקרדיט ל-Felix, Sam ו-Jenny (השמות כפי שנשמעו ב-ASR). לפי Boris, Cowork תפס הרבה יותר מהר משתפס Claude Code [47:47]. משחררים מוקדם מסיבות מוצר וגם מסיבות בטיחות. שכבות הבטיחות שהוא מתאר הן alignment ו-mechanistic interpretability (כאן הוא מזכיר את Chris Olah ואת superposition), אחר כך evals, ואחר כך שימוש בעולם האמיתי [49:17], [52:26]. כחלק מגישת Anthropic race to the top, פרסמה sandbox בקוד פתוח שעובד עם כל agent [53:56].

Boris Cherny [46:13] We said the product is the model. We want to expose it. We want to put the — minimal scaffolding around it, give it the minimal set of tools so it can do the things

7. עקרונות לבניית מוצרי AI וטיפים ל-Claude Code

לבוני מוצרים הוא מציע שלושה עקרונות. הראשון: לא לכלוא את המודל ב-workflow של שלב 1, שלב 2, שלב 3 עם orchestrator מתוחכם, אלא לתת לו כלים ומטרה ולתת לו להשיג לבד את ה-context שהוא צריך [58:38]. השני: ה-bitter lesson של Rich Sutton. המודל הכללי ינצח את הספציפי, ולכן הוא ממליץ לא להשקיע במודלים קטנים או ב-fine-tuning אם יש ברירה. השלישי: לבנות למודל של עוד שישה חודשים, ולא למודל של היום [1:00:10]. לדבריו, לאן המודל ישתפר: שימוש בכלים ובמחשב, וריצה ארוכה יותר. Sonnet 3.5 החזיק 15 עד 30 שניות, ו-Opus 4.6 רץ 10 עד 30 דקות בלי השגחה, ולפעמים שעות, ימים ואף שבועות [1:03:19]. על Codex הוא אומר שכמעט לא ניסה אותו, שהוא נראה לו דומה מאוד ל-Claude Code, ושתחרות טובה [1:06:21].

Boris Cherny [1:00:10] in general, what we see is maybe scaffolding can improve performance, — maybe 10, 20 percent, something like this. But often these gains just get wiped out with the next model

טיפ	מה בדיוק	זמן
המודל החזק ביותר	Opus 4.6 על maximum effort תמיד. מודל זול יותר צריך יותר תיקונים, ולכן לרוב צורך יותר טוקנים	1:04:50
plan mode	כ-80% מהשימות מתחילות בו. זה משפט אחד שמזרק לפרומפט ומבקש לא לכתוב קוד עדיין. בטרמינל: Shift+Tab פעמיים	1:04:50
auto-accept אחרי תוכנית	כשהתוכנית טובה, Opus 4.6 לרוב מבצע אותה נכון בניסיון הראשון	1:04:50
לנסות ממשקים שונים	טרמינל, דסקטופ, iOS ו-Android, ווב, Slack. בכולם רץ אותו agent	1:06:21
כמה sessions במקביל	multi-clauding: הרבה sessions של Claude במקביל, למשל בלשונית Code בדסקטופ	41:39
Cowork בשלבים	להתחיל משימה עם כלי אחד, אחר כך לחבר כמה כלים (מייל ל-Slack או לגיליון), ואחר כך אוטומציות קבועות כמו תזכורת סטטוס בכל יום שני	1:17:07

8. מה עושים עם זה

- **לבדוק מחדש הנחות ישנות:** לפני שפותרים בעיה ידנית, לנסות קודם לתת אותה ל-Claude Code. סיפור דליפת הזיכרון מראה שגם מומחים נתקעים עם הרגלים מימי Sonnet 3.5.
- **לפרק workflows נוקשים:** לעבור על pipelines של LLM שבנויים כרצף שלבים קשיח, ולבדוק אם agent עם כלים ומטרה מגיע לאותה תוצאה. רוב ה-scaffolding צפוי להתיישן עם המודל הבא.
- **לבצע benchmark על עלות לפי משימה:** להשוות את המודל החזק ביותר מול מודל זול לפי עלות כוללת למשימה שהושלמה, ולא לפי מחיר לטוקן.
- **להפוך את plan mode לברירת מחדל:** להתחיל במשימות לא טריוויאליות עם תוכנית, ואז לבצע עם auto-accept.
- **לחפש latent demand:** לבדוק איפה משתמשים ובפרט המודל מנצלים את המוצר למשהו שהוא לא תוכנן בשבילו.
- **ללולאת הפידבק:** לחבר את Claude לערוץ הפידבק הפנימי ולתת לו להציע PRs. תגובה מהירה גורמת לאנשים לשלוח עוד פידבק.

שאלה פתוחה	למה זה חשוב	מה נאמר בשיחה
מה איכות הקוד שנוצר על ידי AI לאורך זמן?	אם 100% מהקוד נוצר על ידי AI, קשה לדעת מה קורה ל-tech debt ולבאגים	לא נמסרו נתונים. נאמר רק שעדיין צריך review אנושי
מתי עלות הטוקנים עולה על הערך שהם מייצרים?	מהנדסים שמוציאים מאות אלפים בחודש	רק להשאיר אופטימיזציה לשלב מאוחר
האם 50% חפיפה בין תפקידים תוביל למיזוג שלהם?	מבנה צוותים של PM, מעצב ומהנדס	בטווח הקצר החלוקה נשארת, ולטווח הארוך זה לא ברור
למה מעצבים נהנים פחות?	55% מול 70% בסקר של המגיש	לא נמצא הסבר
מתי באמת לא צריך להבין את השכבה שמתחת?	תכנון הכשרה והעסקה של מהנדסים צעירים	לדבריו בעוד כשנה, וזו הערכה בלבד

9. מילון מונחים

מונח	משמעות
agent	מודל שפה שיכול להפעיל כלים ולפעול בסביבה, ולא רק לנהל שיחה
Cowork	מוצר של Anthropic בתוך אפליקציית הדסקטופ שמריץ את Claude Code למשימות לא טכניות, כולל שליטה ב-Chrome
latent demand	ביקוש חבוי: שימוש במוצר למטרה שלא תוכננה, שמאותת לאן לפתח אותו

מונח	משמעות
on distribution	לבנות בהתאם למה שהמודל נוטה לעשות באופן טבעי
scaffolding	קוד מעטפת, workflows ו-orchestration שעוטפים את המודל
bitter lesson	הרעיון של Rich Sutton שהגישה הכללית מנצחת בטווח הארוך את הגישה הספציפית
plan mode	מצב שבו המודל מתכנן ולא כותב קוד עד שהתוכנית מאושרת
multi-clauding	הרצה של כמה sessions של Claude במקביל
mechanistic interpretability	חקר הנירונים של המודל כדי להבין מה הם מייצגים
superposition	מצב שבו נירון אחד מייצג כמה מושגים בשילוב עם נירונים אחרים
evals	מבחנים מבוקרים להתנהגות המודל
race to the top	הגישה של Anthropic: לפרסם כלים ומחקר בטיחות כדי שגם מעבדות אחרות יאמצו אותם
sandbox	סביבת ריצה מבודדת שמגבילה את הגישה של agent למערכת
heap snapshot	תמונת מצב של הזיכרון שמשמשת לאיתור דליפות
DAU	מספר המשתמשים הפעילים ביום
MCP	Model Context Protocol: פרוטוקול לחיבור כלים למודלים, שנבנה באותו צוות