==========
התקנה והגדרת **Hardhat**
==========

מה זה **Hardhat?**

****Hardhat****
הוא סביבה לפיתוח חוזים חכמים ב-**Ethereum**.
הוא מאפשר כתיבה, קומפילציה, בדיקה ופריסה של חוזים בצורה פשוטה ונוחה.
Hardhat
כולל כלים רבי עוצמה לניהול פרויקטים מבוססי **Solidity**.

שלבי התקנה והגדרת **Hardhat**
---

npm install --save-dev hardhat

כדי ליצור את הפרויקט לדוגמה, הפעל
npx hardhat init

כדי לקבל תחילה תחושה מהירה של מה זמין ומה קורה, הפעל
npx hardhat

תראה contracts/, לאחר מכן, אם תסתכל בתיקייה
Lock.sol:

כדי לקמפל אותו, פשוט הפעל:

npx hardhat compile

אם תסתכל בתיקייה
test/,
תראה קובץ בדיקה

אתה יכול להריץ את הבדיקות שלך עם

npx hardhat test

בתוך ignition/modules

התיקיה תמצא קובץ עם קוד:

אתה יכול לפרוס אותו באמצעות
npx hardhat ignition deploy ./ignition/modules/Lock.ts

Hardhat Network ייצור מופע חדש בזיכרון של Hardhat, כברירת מחדל

בעת ההפעלה.

אפשר גם להפעיל את

Hardhat Network

בצורה עצמאית כך שלקוחות חיצוניים יוכלו להתחבר אליה.

זה יכול להיות ארנק,

שלך Dapp-חזית ה

או פריסת Hardhat Ignition.

בדרך זו Hardhat Network כדי להפעיל את,

הפעל npx hardhat node:

Hardhat לרשת JSON-RPC זה יחשוף ממשק

כדי להשתמש בו חבר את הארנק או האפליקציה שלך ל http://
127.0.0.1:8545.

אם אתה רוצה לחבר את

לצומת זה, למשל כדי להפעיל נגדו פריסה Hardhat,

אתה פשוט צריך להפעיל אותו באמצעות

--network localhost.

כדי לנסות זאת, התחל צומת עם

npx hardhat node

האפשרותnetwork והפעל מחדש את הפריסה באמצעות:

npx hardhat ignition deploy ./ignition/modules/Lock.ts --network localhost
ביציאה ספציפית ולאפשר בקשות Hardhat Network כדי להפעיל את נכנסות ממשק רשת או שם מארח ספציפיים, הפעל

npx hardhat node --hostname 127.0.0.1 --port 8545.

אם ברצונך לאפשר בקשות נכנסות מכל מקום, כולל כתובות

חיצוניות IP,

השתמש ב- --hostname 0.0.0.0.

==========
מבנה תיקיות וקבצים בפרויקט **Hardhat**
==========

כאשר יוצרים פרויקט **Hardhat**, מתקבלות תיקיות וקבצים שמטרתם לספק תשתית מסודרת לפיתוח חוזים חכמים. להלן פירוט מלא על כל תיקייה וקובץ.

תיקיות עיקריות

---

artifacts

תיקייה זו נוצרת לאחר תהליך הקומפילציה (הפיכת הקוד ב-**Solidity** לקוד מובן על ידי הבלוקצ'יין).

היא מכילה:

- build-info:

קבצים עם מידע על תהליך הקומפילציה. נועדו בעיקר למעקב אחרי גרסאות החוזים.

- contracts:

קבצים שנוצרים לכל חוזה חכם שקומפלת.

לכל חוזה חכם יהיו שני קבצים:

- JSON שמכיל את ה-**ABI (Application Binary Interface)** ואת הקוד לפריסה.

- קובץ **debug.json** למעקב ובדיקות דיבוג.

cache

תיקייה זו משמשת לאחסון זמני של קבצים שקשורים לקומפילציה.

לדוגמה:

- solidity-files-cache.json:

מטמון שמאפשר ל-**Hardhat** לזהות אילו חוזים כבר קומפלו, כדי לחסוך זמן.

contracts

התיקייה החשובה ביותר לפיתוח!

כאן תכתוב את החוזים החכמים שלך בשפת **Solidity**.

לדוגמה:

- Lock.sol:

חוזה לדוגמה שנוצר עם הפרויקט. זהו חוזה פשוט שמדגים נעילת כספים עד לזמן מסוים.

explanations

תיקייה זו משמשת לאחסון קבצים שקשורים לתיעוד, הסברים או תוספות לפרויקט.

לדוגמה:

- hardhat.rst:
קובץ **RST** שמסביר על הפרויקט או תהליכים בו.

ignition
תיקייה זו מאפשרת יצירת מודולים מותאמים אישית לפריסה מתקדמת של חוזים.
לדוגמה:
- modules/Lock.ts:
מודול לדוגמה שמכיל סקריפט פריסה.

test
תיקייה זו מיועדת לכתיבת בדיקות לחוזים החכמים שלך.
הבדיקות מבוצעות אוטומטית כדי לוודא שהחוזים עובדים כראוי.
לדוגמה:
- Lock.ts:
קובץ בדיקות לדוגמה עבור חוזה **Lock**.

typechain-types
תיקייה זו נוצרת אוטומטית כאשר אתה משתמש ב-**TypeChain** (ספרייה שמתממשקת עם **TypeScript**).
היא כוללת מחלקות ופונקציות שעוזרות לך לתקשר עם החוזים שלך בצורה בטוחה מבחינת טיפוסים.

קבצים חשובים בפרויקט

README.md
קובץ תיעוד ראשי של הפרויקט. נועד להסביר בקצרה על מטרת הפרויקט והשלבים שבו.

hardhat.config.ts
קובץ ההגדרות של **Hardhat**.
בתוכו תגדיר:
- רשתות בלוקצ'יין (לוקאליות, טסטנט או פרודקשן).
- פלאגינים נוספים.
- נתיבים מותאמים אישית.

package.json

קובץ המנהל את התלויות **(dependencies)** של הפרויקט.
הוא מגדיר:
- את הספריות שבהן הפרויקט משתמש.
- סקריפטים שניתן להריץ (כמו קומפילציה או פריסה).

package-lock.json

קובץ זה מוודא שהתלויות נשארות זהות בכל התקנה מחדש, כדי למנוע בעיות תאימות.

tsconfig.json

קובץ הגדרות של **TypeScript**.
הוא מגדיר כיצד הקוד יומר מקוד **TypeScript** לקוד **JavaScript**.

סיכום

בפרויקט **Hardhat** יש מבנה מסודר שמחלק את העבודה לכמה שלבים:

1. כתיבת חוזים בתיקיית **contracts**.
2. בדיקות בתיקיית **test**.
3. פריסה בתיקיית **ignition** או **scripts**.
4. קומפילציה וניהול נתונים בתיקיות **artifacts** ו-**cache**.