**🎯 What is Pump.fun?**

Pump.fun is a platform on the **Solana blockchain** that lets **anyone create and trade meme coins** easily and instantly — no coding or deep crypto knowledge needed.

### 🛠️ ****How it Works (Step by Step):****

#### 1. **Create a Token (Meme Coin):**

* You click a button to “Create a coin”.
* Pump.fun automatically:
  + Creates your token (like $MEME).
  + Locks 100% of the supply in a **liquidity pool**.
  + Sets up a **bonding curve** (more on that below).
  + Gives you a link to share your coin with others.

#### 2. **Bonding Curve Pricing:**

* The price of the token increases automatically as more people buy it.
* This is based on a **bonding curve**, a math formula that raises the price as demand increases.
* So:
  + **Early buyers get cheap prices**.
  + **Late buyers pay more**.

#### 3. **Buying/Selling Coins:**

* Anyone can buy/sell your coin using SOL (Solana’s token).
* Everything happens directly on-chain (no middlemen).
* When someone buys, they push the price up (because of the curve).
* When they sell, they get SOL back and the price drops.

#### 4. **Pump Score & Leaderboard:**

* Coins are ranked by their **"Pump Score"** (based on price increase and volume).
* Top trending coins get visibility on the homepage, so it creates hype.

**💧 What is Liquidity (in crypto, simply explained):**

**Liquidity** means **how easily you can buy or sell a token** without affecting its price too much.

**🔄 Example:**

Imagine a water tank:

* If the tank is full (lots of water = **high liquidity**), you can take water in or out without a problem.
* If the tank has just a little water (low liquidity), even a small change (someone taking water) makes a big impact.

Same with crypto:

* If a token has **high liquidity**, you can buy or sell it quickly, at a stable price.
* If it has **low liquidity**, buying or selling causes **big price swings**, and it's harder to trade.

**🪙 What is an LP Token (Liquidity Provider Token)?**

An **LP token** is like a **receipt** you get when you add liquidity (tokens) into a **liquidity pool** on a decentralized exchange (DEX), like Uniswap, Raydium, or Orca.

**🧃 Think of it like this:**

1. You put your **juice** (SOL + MemeCoin) into a big punch bowl (liquidity pool) at a party.
2. The party host gives you a **stamp** (LP token) to prove **how much juice you added**.
3. Later, when you want your share of the punch back — plus some **party tips** (trading fees) — you return the stamp.

## **🛠️ What is Hardhat?**

**Hardhat** is a **development environment for Ethereum smart contracts**.

Think of it like a toolkit or IDE (like VS Code, but for blockchain devs) that helps you:

* Write smart contracts (usually in **Solidity**).
* Test them locally.
* Deploy them to blockchains (like Ethereum, Polygon, etc.).
* Debug and automate things.

## 💰 What is **wei**?

**Wei** is the **smallest unit of Ether (ETH)** — like a **paisa** or **cent** in Ethereum.

### 🧮 Conversion:

markdown

1 ETH = 1,000,000,000,000,000,000 wei

= 10^18 wei

So 1 wei is **0.000000000000000001 ETH**

### 📦 Why do we use wei?

* Because Ethereum is super precise.
* Gas fees and smart contracts often deal with **tiny amounts**.
* Just like Bitcoin uses **satoshis**, Ethereum uses **wei**.

## 🍵 What is **Chai**?

**Chai** is a **JavaScript assertion library** used for **testing**.

Think of it like a **tool that checks if your code is doing what it’s supposed to do**.

## 🧪 Where is it used?

Chai is commonly used with:

* **Mocha** (a test framework)
* **Hardhat** (for testing smart contracts)

## 📦 What is describe() and it()?

They’re functions used to **organize and run tests**.

### 🧪 describe() = A group of related tests

It’s like a **folder** or **section** for tests.

js

CopyEdit

describe("Token Contract", function () {

// test cases go here

});

### 🔍 it() = A single test case

### It’s like saying: “It should do something...”

it("should return the correct name", function () {

// test logic here

});

## 📄 Example (Full Context):

## 

## 🪙 What is **mint** in crypto?

**Minting** means **creating new tokens** and adding them to someone’s wallet.

### 🧠 Think of it like:

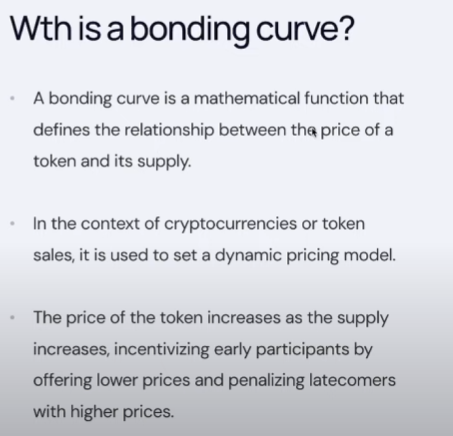
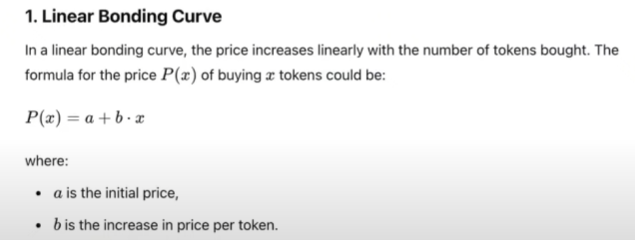
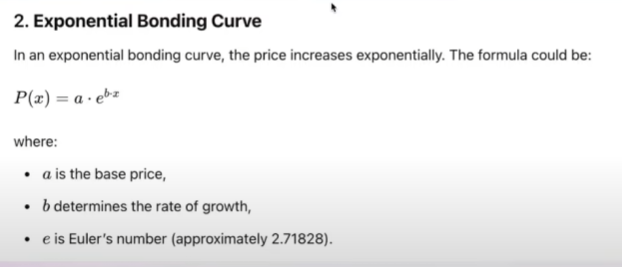
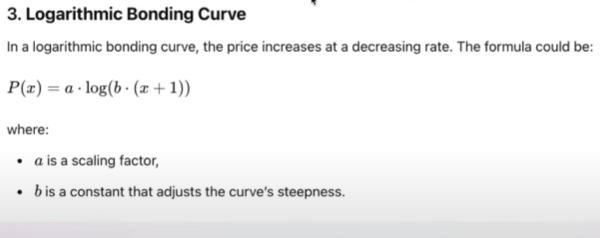
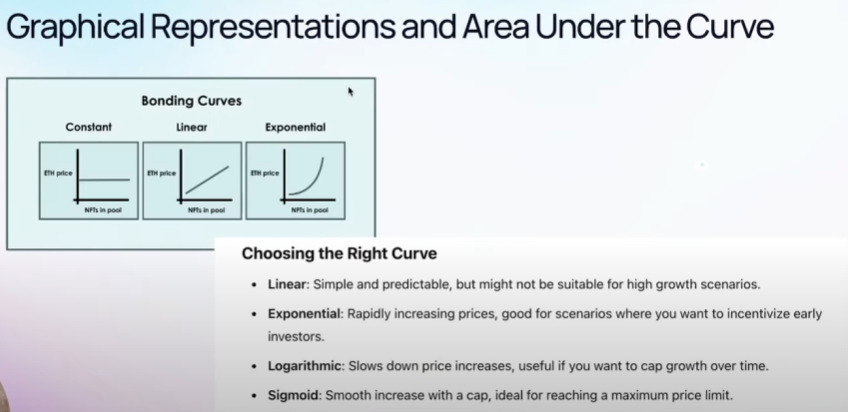
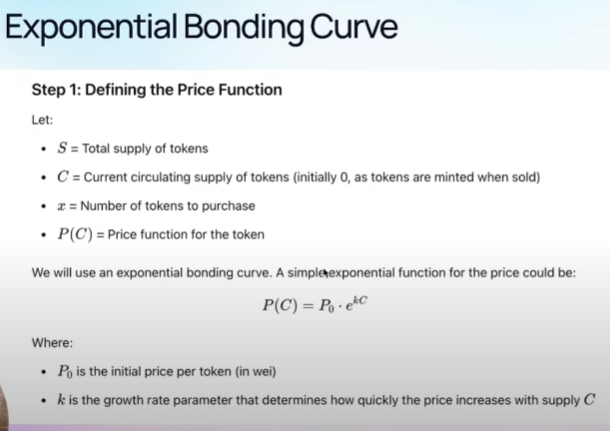
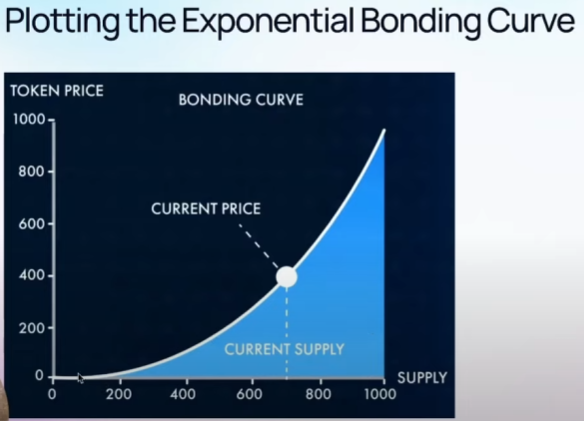
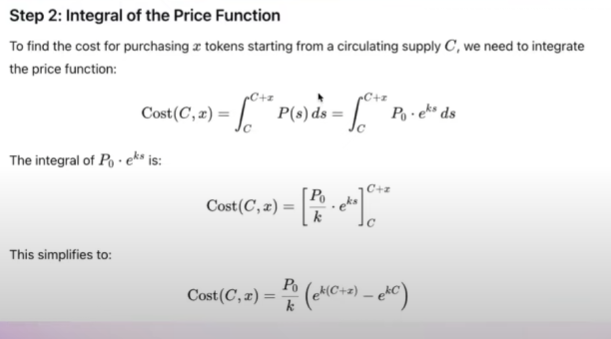
* Minting = 🖨️ Printing new money (but on the blockchain)
* You’re **bringing tokens into existence** that didn’t exist before



***Homer.fun (ReactJS, Solidity, hardhat, Moralis)***

* Building on EVM chain. Which means the token we are created a ERC20 token.
* Mechanics used 🡪 pump.fun
* Each token a user created has maximum supply, 20% reserved if the funding goal is met.
* In this project we are going to set the funding goal 🡪 24ETH.
* When the market cap reaches 24ETH, all the liquidity from the bonding curve will be deposited into Uniswap, and the LP tokens will be burned.
* npm init -y
* npx hardhat init
* npm install @openzeppelin/contracts
* npm install @uniswap/v2-core @uniswap/v2-periphery
* Testing of hardhat env🡪npx hardhat test test/TokenFactory.js

(default 🡪 hardhat network env)

Uniswap

* npm i --save @uniswap/v2-core
* npm i --save @uniswap/v2-periphery

**note:-** currently we are running on test network Sepolia, in order to completely shift to the mainnet, we need to change the uniswap contact deployment address from the TokenFactory.sol (<https://docs.uniswap.org/contracts/v2/reference/smart-contracts/v2-deployments>) and change the Moralis API key from in .env file (<https://admin.moralis.com/nodes>).

* Make sure to change in hardhat.config.js file
* Testing 🡪