Why Gno? The Need for the GNO VM for Smart Contract Programming

Gno is the easiest and most intuitive smart contract programming language available today. Gno contracts are more succinct and easier to audit than Solidity and Cosmos SDK ones. Gno lowers the barrier for new Web3 developers and makes it trivial for existing Golang devs to start writing Web3 smart contracts.

With its simplicity, transparency, gentle learning curve (even for newbies) and scalability, Gno is ripe for rapid adoption. Gno and the Gno VM are future-proof, with straightforward interoperability and massive concurrency.

Gno looks forward to a Web3 beyond DeFi that focuses on social coordination dapps. Gno and the Gno VM exist so that people everywhere can build, collaborate, discover truth and communicate without censorship.

Whether you're already a smart contract developer or want to learn, you'll want to know more about Gno. We'll start by discussing Gno's top 8 benefits and finally provide some resources for getting started with Gno.

1. Gno is Intuitive

Gno is an easy language to understand at a glance, given its succinct logic. Gno only has 25 keywords, so you can learn them in no time at all and be sure your contracts will work as expected.

Gno contracts are much shorter than comparable ones written in Solidity or with the Cosmos SDK. For example, here is the simplest smart contract you can write in Gno.

```
var x int
```

```
func Incr() {

x += 1

}
```

2. Gno is Auto-Persisted

With Gno, there is no need to move data into and out of databases, unlike with the Cosmos SDK. The current state of your dapp is persisted in memory (unless garbage-collected) and auto-merklized based on the data structures you define.

This new paradigm is a unique feature of the Gno VM that enables the state of any Gno dapp to be verified at any time via Merkle proof — even from other chains.

3. Gno Makes You More Productive

Build your smart contracts faster with Gno. Smart contracts written in Gno are simpler and clearer. Even a newbie can start building right away.

Accelerate the building of your complex dapps by leveraging Golang's extensive range of packages and tooling. If you're already a Golang developer, then building your smart contracts with Gno is a no-brainer, as you have almost no learning curve at all.

Gno smart contracts are memory-safe so garbage collection is taken care of for you. When deployed, you also get automatic tests and documentation for your code.

4. Gno is Primed for Adoption

Gno contracts are easier to read, write, audit and learn from than Solidity or Cosmos SDK ones. Due to its simplicity, Gno will enable more people to build their first Web3 smart contracts. Gno's modularity and composability also make creating complex dapps easier.

Gno is 99% identical to Golang. Gno builds on Golang's existing superpowers, including efficient concurrency and access to many existing Golang libraries. This makes Gno a powerful tool for developers from day one.

Learn Gno once and you can understand the entire Gno stack, as it's all written in Gno. This, as opposed to Solidity, where you need to be comfortable with multiple languages to understand the whole stack or work in multiple ecosystems. You can import packages in Gno and benefit from the same constructs as in Solidity.

Millions of Web2 Golang developers can easily adapt their current web apps to Gno or build new Web3 ones. This dramatically lowers the barrier to entry for Web3 development.

5. Gno is Scalable

Gno-based blockchains are more scalable due to the Gno focus on simplicity, efficient execution, speed, concurrency and long-term sustainability.

Most other blockchains can only process one transaction at a time. With Gno, however, you can break your dapps into smaller chunks that can run simultaneously with minimum overhead and full utilization of the available CPU cores.

Gno also takes advantage of the cost-effective AVL tree data structure for large datasets. AVL trees are self-balancing binary search trees that load only the data you need when you need it, thus saving on gas fees and facilitating scaling.

6. Gno is Easy to Audit

Gno is interpreted, not compiled. So, unlike other smart contract languages, such as Solidity, there is no hard-to-understand bytecode version. Your Gno source is what you upload to the chain.

The Gno VM uses the Golang abstract syntax tree (AST) — if, else, func, struct, etc. — instead of compiling to bytecode. Everything is written in Golang itself. This makes Gno code quite simple and easy to read.

In this way, transparency is a fundamental feature of the system, and is not optional. This minimizes the need for third-party auditors to verify that the bytecode matches the published source code, unlike with Solidity.

You can also leverage other developers' packages on the Gno.land hub blockchain to get peace of mind, knowing that you're building on other developers' reputations. It is trivially easy to verify existing contract packages or fork them into a version that you can improve upon.

7. Gno is Interoperable

With Gno's auto-persistence, interoperability between chains is easier than ever before. The IBC (Inter-Blockchain Communication Protocol) can be built right into your smart contract. Version 2 of the IBC will likely be an interchain protocol that just reads the current state of an on-chain variable — and proves it with a Merkle proof.

Gno smart contracts themselves are also highly interoperable. The composable nature of Golang and Gno gives developers the ability to create complex contracts by combining smaller, reusable packages. The Gno.land chain will be a GitHub-like central repository for Gno code where you can do just this.

Embed structs to create more complex structures. Assign functions to variables, pass them as arguments and return them from other functions to create higher-order functions. And it's all type-checked to catch errors before contract execution and optimize your code for performance.

8. Gno is Secure

Unlike other virtual machines, such as WebAssembly, Gno is deterministic. Determinism is crucial for smart contracts because it guarantees that all network validators will reach the same state after executing a given contract.

Gno also benefits from all of Golang's built-in security features, including the "least-authority" design principle. This enables you to control which objects and APIs are exposed to the public solely based on the capitalization of letters in your code.

Get Started with Gno Today

The forthcoming **Gno.land** blockchain will be based on Tendermint2 and leverage Gno and the Gno VM. Gno.land will also have a novel consensus mechanism called Proof of Contribution. By making meaningful contributions to the Gno.land ecosystem, you can earn contributor tokens. Eventually you may rise up the ranks to become a chain validator and earn rewards that pay out on a recurring basis over time.

Web3 is about challenging centrally controlled systems and stagnant institutions. Gno and the Gno VM are precisely the tools we need to distill the truth in a truly unstoppable network that is accessible and transparent to all.

Try out the **Gno Playground** today, a web-based Gno IDE where you can write, share and deploy Gno code. Or get Gno support in ViM, Emacs or **Visual Studio Code**. Build your first Gno dapp with the **Getting Started repo** or **learn Gno by example**. Build a new project, and apply to the Gno.land **grants program**.

Whether you're a seasoned smart contract developer or just starting out, Gno and the Gno VM provide the simplicity, security, scalability, and interoperability you need to build the decentralized applications that will power the Web3 revolution.