🚀 Effective Gno

Gno.land Development Best Practices

📦 Prefer Small Dependencies

- Unlike Go's "little copying" philosophy

- Reuse audited, community-vetted packages

- Builds trust through composability:

```
import (
    "gno.land/p/finance/tokens"
    "gno.land/p/utils/permissions"
)
```

📖 Documentation is for Users

- Write for end-users, not just developers

- Include clear endpoint explanations

```
// Transfer moves tokens between accounts
// - to: recipient address
// - amount: positive integer
func Transfer(to std.Address, amount int) {
    // ...
}
```

🧱 Design Realms as Public APIs

- Treat all exposed functions as public endpoints

- Validate all inputs

- Use private functions for internal logic

```
func PublicMethod() {
    validateCaller()
    privateLogic()
}
```

🔐 Security & Access Control

🛡️ Construct "Safe" Objects

- Objects that enforce their own security

- Protect internal state

- Can be shared between realms

```go
type SafeVault struct {
    balance int
    owner   std.Address
}

func (v *SafeVault) Withdraw(amount int) {
    if std.PreviousRealm() ≠ v.owner {
        panic("unauthorized")
    }
    // ...
}
```

🌳 Use avl.Tree for Large Datasets

- Efficient key-value storage

- Lazy loading reduces gas costs

- Ideal for large collections

```go
import "avl"

var users avl.Tree

func GetUser(id string) *User {
    return users.Get(id).(*User)
}
```

💾 State Management

📡 Emit Events for Off-chain Use

- Indexable by external services

- Key-value format for filtering

```
func Transfer(to std.Address, amount int) {
    std.Emit("Transfer",
        "from", std.PreviousRealm().String(),
        "to", to.String(),
        "amount", strconv.Itoa(amount))
}
```

💰 Token Strategies

⚖️ Coins vs GRC20 Tokens

Feature	Coins	GRC20 Tokens
Management	Banker module	Smart contract
IBC Support	✅ Yes	❌ Not yet
Flexibility	Limited	High (custom logic)
Gas Costs	Lower	Higher
Use Cases	Native currency, IBC	DeFi, DAOs, token-gating

🎁 Wrapping Coins

```go
import "gno.land/p/demo/grc20"

// Wrap native coins into GRC20-compatible token
var wrappedCoin = grc20.NewBanker("Wrapped Coin", "WRAP", 6)
```

📦 Package Structure Best Practices

📜 Key Rules

1. Match package name to directory name

2. Define interfaces and types in `p/`

3. Keep realms in `r/` focused on business logic

4. Use `internal/` for private packages

5. Prefer small, focused packages

🏁 Conclusion

- ✅ Embrace Gno-specific patterns

- ✅ Prioritize security and access control

- ✅ Design for users and composability

- ✅ Choose appropriate storage solutions