导论

在广泛分析诸如 Solidity Github 仓库、 Solidity 发展路线图、 Twitter 上的社群对话、活跃的 Pull Requests 和 Issues 等信息后，本文深入探讨了 Solidity 的未来将会走向何方。

这门领先的智能合约编程语言即将发布 0.9.0 和 1.0.0 升级，其中将推出几项备受期待的增强功能。

本文旨在向读者介绍Solidity的最新发展和改进，这些发展和改进基于社区的投入和正在进行的辩论。虽然提供的相关信息尚无定论，但它揭示了潜在的技术进步方向。

1. 革命性地将 `require()` 与 Custom Error 进行集成

当前方法（0.8.x）：

```
error UnauthorizedAccess();

if (condition) {
  revert(UnauthorizedAccess());
}
```

预期将在 0.9.0 或 1.0.0 出现：

```
error UnauthorizedAccess();

...

require(condition, UnauthorizedAccess());
```

相比于使用大量的 **if** 条件用**字符串信息**或 **custom error** 抛出错误，将 **require()** 和 custom error 结合起来使用将使程序更加清晰并节省 gas 。

2. 内部表示（IR）最优化：释放效率

Solidity 中的**内部表示**（IR）过程在将智能合约源代码转换为以太坊虚拟机（EVM）的可执行指令方面发挥着关键作用。

内部表示**令复杂的代码简洁化与标准化**，使得将代码转换为机器语言的过程更加高效。 Solidity 即将推出的改进 0.9.0 旨在使这种编译过程更快、更高效，**最终降低成本并提高开发人员的生产力。**

3. 增强错误处理：澄清和简化代码

预计未来的更新将增强 Solidity 中的错误处理，从而导致更直接的错误消息和更轻松的调试。

这种增强对于处理复杂合约的开发人员特别有用，可以节省时间并最大限度地减少潜在错误。

4. 定点数算数：高精度与高性能

【译者注：定点数是与浮点数相对应的计算机用于表示小数的数据类型，其整数位与小数位长度均为定值，因此比浮点数更加安全。目前的 Solidity 并没有内置任何表示小数的数据类型，开发人员常常需要将原始数据乘以一个很大的数来手动预留出小数位，并在之后的运算中时刻留意小数点问题，相当麻烦。】

现阶段，诸如 ABDKMath64x64 和 DSMath 的外部库都用 Solidity 实现了定点数。

0.9.0 版本的更新预计将**集成原生的定点数运算**，不再需要外部库。这将使小数计算更加简化。

5. EVM 对象格式（EOF）：构建智能合约字节码

Solidity 即将到来的 EOF 升级准备为智能合约引入结构化的和版本控制的字节码。

这一改进有望使将来的合约更新升级更加容易实现，保持向后兼容性，并在编译阶段实现更有效的分析。

虽然这不会改变智能合约开发人员的直接编码体验，但**编译器的输出将会更省 gas** 。

6. 瞬态存储：临时而高效的数据处理

Solidity 的新功能，瞬态存储，提出了一种在合约执行期间临时保存数据而不将其永久记录在区块链上的方法。这种方法预计在 gas 消耗方面更有效率。[点击了解更多关于瞬态存储的信息]

类似于如下代码的瞬态存储很有可能出现在 Solidity 0.9.0 或者 Solidity 1.0.0 。

```
uint transient x;
```

7. 原生集成重入保护

在版本 0.8.0 之前，** SafeMath 库**是开发人员用于算术运算以避免溢出和下限溢位问题的常用工具。随着 Solidity 0.8.0 的发布，这些安全检查被直接嵌入到语言中。

类似地， Solidity 版本 0.9.0 或 1.0.0 预计将**原生集成重入保护**。此功能旨在简化实施安全协议以防止重入攻击的过程。

你将会看到类似这样的东西：

```
function myReentrantFunction(uint x, uint y) external reentrant returns (uint)
```

8. 重构继承和存储（Storage）布局

在Solidity中，继承可以创建一个新合约，该合约采用现有合约的属性和功能。计划中的更新旨在**改进继承和存储布局的线性化**，从而增强合约架构的可预测性和组织性。**这可以提高存储使用效率，减少多继承场景中的混乱。**

> 例如，一个继承自 `ParentA` 和 `ParentB` 的合约 `Child` 将会有最优化的存储布局和连续的变量存储，降低存储操作的花费。

9. 增强的编译标志和配置选项

Solidity 的开发包括扩展编译标志和配置设置的范围，为开发人员提供对编译过程的高度控制。这些增强可能会导致更加定制化的合约部署，对 gas optimization、安全检查和调试功能等方面进行精细控制。

新的编译标志可以允许开发人员切换特定的优化或安全检查：

> 例如，一个新的编译标志 `--enable-loop-optimization` 将会专注于优化循环来提高 gas efficiency，另一个新的编译标志 `--strict-security-checks` 将会在编译过程中引入严苛的安全分析

10. 改进的调试工具和错误信息

强化后的调试工具具有更翔实的错误信息，可以显著简化开发过程，尤其是对于复杂的合约结构。改进的错误信息可以更促进开发人员深入理解代码中的问题，而高级调试工具可以帮助开发人员更有效地发现和修复问题。

11. 支持高级数据类型和结构

在 Solidity 中引入复杂的数据类型和结构可以启发新的合约设计和功能可能性。这可能包括支持更复杂的数字类型、增强的数据结构或改进处理合约中大型数据集的方法。

Solidity 可能会引入一种新的数据结构，如 `TreeMap`，它以排序的方式组织数据，从而实现高效检索。这可能在需要对数据进行排序或排序的合约中很有用，比如在投票系统中。【译者注：类似于红黑树。】另一个进步可能是支持更复杂的数字类型，如定点数，可以直接在合约中进行精确的数学运算。

12. 引入泛型和模板

Solidity 中的泛型和模板将使得更具适应性和可重用性的代码成为可能。例如，可以创建一个泛型函数来以标准化的方式处理不同类型的资产（如 ERC20 代币、 NFT 等），而无需为每种特定的资产类型重写函数。这将改进合约设计方式和开发效率，因为单个函数可以应用于各种场景。

```
contract AssetHandler<T> {
  T asset;

  function processAsset(T _asset) public {
    // Generic processing logic for various asset types
  }
}
```

未来展望：Solidity 1.0.0 之路

在诸如 Github 、 Twitter 、 Ethresearch 和 Reddit 的各种平台上， Solidity 社群内部关于 0.9.0 版本发展规划的讨论如火如荼。

一个焦点性的争论正在揭晓：

> 是直接谨慎地过渡到 Solidity 1.0.0 以宣告这门语言的完全成熟，还是首先通过 0.9.0 版本逐渐地进步到更高级的版本？

受社区反馈和创意想法影响， Solidity 1.0.0 的预期首次亮相可能与以太坊的重大更新保持一致，反映了整个生态系统的增长和稳定。

1. **类型系统的演变**：提高灵活性和安全性。预计会有一个升级类型的系统，从 Haskell 或 Scala 等函数式编程语言中汲取灵感。这种演变旨在增合约开发的安全性和灵活性。
2. **集成原生预言机支持**：简化外部数据交互。计划包括在 Solidity 中整合对去中心化预言机的内置支持，促进与外部数据源的更安全和直接的交互。
3. **改进状态管理**：完善区块链交互。状态管理能力的增强正在讨论中，可能会引入状态通道或侧链等元素作为内置结构，旨在优化区块链上的交互并降低 gas 费用。
4. **合约设计中的模块化方法**：提高重用性。人们正在设想向模块化合约架构的转变，允许使用可替换的组件。这可以显着简化开发过程并提高代码的可运维性。
5. **集成形式化验证工具**：确保合约可靠性。人们期望将形式化验证工具直接集成到 Solidity 中，这一举措旨在确保合约符合特定的标准和行为，从而降低错误和漏洞的可能性。
6. **建立跨链能力**：实现跨区块链的互操作性。未来的更新可能会引入原生的跨链兼容性功能，从而允许 Solidity 合约在各种区块链协议中顺利运行。
7. **实施高级隐私措施**：增强数据安全。人们正计划将高级隐私工具（如零知识证明或同态加密）直接整合到语言中，旨在加强数据安全和用户隐私。
8. **抗量子密码学**：为未来的挑战做准备。鉴于新兴的量子计算能力，人们正在考虑实施抗量子加密方法以保护以太坊合约免受未来潜在的威胁。

结论

在探索 Solidity 的潜在路径时，本文将社区见解和当前发展与社区预测相结合，让您对主题有一个全面的理解。虽然我们已经深入研究了 0.9.0 和 1.0.0 版本的可能性，但随着 Solidity 继续其开发之旅，实际的轨迹和功能集可能会发生变化。敬请关注这些对话和概念如何在不断发展的智能合约编程世界中实现。

更多信息

作者简介：Adam Boudjemaa，Biconomy🍊区块链技术主管，去中心化金融创新者， ERC 标准贡献者，擅长智能合约开发和 EVM 优化。

译者简介：怀老师，武汉大学 Web3 俱乐部投研部部长，本科在读，主要研究安全审计、零知识证明和 DeFi 衍生品。

原文链接：https://medium.com/solichain/solidity-1-0-0-next-gen-smart-contracts-c5a94e3f72d6