

# Solidity Advanced (三)

Constructor 与 Contract Inheritance · 小学生也能懂

---

## Constructor (建構子)

### 一句话给小学生

Constructor 是：合约出生时只跑一次的函数。

### Constructor 的特性

- 只在「部署合约」时执行
- 一辈子只跑一次
- 不能被别人再呼叫
- 常用来设定初始资料

### Constructor 範例 (中文注释)

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.10;

contract X {
    string public name;

    // 部署合约时，必须传入一个字串
    constructor(string memory _name) {
        // 合约一出生，name 就被设定好
        name = _name;
    }
}
```

### 小学生重点

- Constructor 不是普通函数
- 不能再呼叫第二次
- 常用来设定 owner、名称、初始值

## Contract Inheritance (合约继承)

### 一句话给小学生

继承 = 小合约可以用大合约的功能。

### Solidity 的继承规则

- 使用 `is` 关键字
- 支援多重继承
- 父合约的函数要被改，必须是 `virtual`
- 子合约改函数要写 `override`

## 最基本的继承方向（一定要搞懂）

### 这句话谁对？

- `contract B is A {...}`

正确理解是：

- B 继承 A
- A 是父合约 (Parent)
- B 是子合约 (Child)

错误理解（不要这样想）：

- A 继承 B

### 记忆口诀

`is` 的右边，是爸爸。

## 虚拟函数与覆写 (virtual / override)

### 多重继承完整示例 (中文注释)

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.10;

/*
继承关系图：
    A
   / \
  B   C
  \ /
    D
*/
contract A {
    // virtual: 允许子合约改写
    function foo() public pure virtual returns (string memory) {
        return "A";
    }
}

contract B is A {
    // override: 改写 A.foo()
    // virtual: 允许更下面的合约再改
    function foo() public pure virtual override returns (string memory) {
        return "B";
    }
}

contract C is A {
    function foo() public pure virtual override returns (string memory) {
        return "C";
    }
}

contract D is B, C {
    // override(B, C): 表示两个父合约都有 foo()
    function foo() public pure override(B, C) returns (string memory) {
        // super 会依照继承顺序往右找
        return super.foo();
    }
}
```

## 继承顺序很重要（右边赢）

### Solidity 的规则

当多个父合约有同名函数时：

- 最右边的父合约优先

### 小学生版理解

- 爸爸很多个
- 排在最右边的说了算

## super 是什么？

### 一句话

super = 帮我叫爸爸的函数

### super 的行为

- 不是固定某一个父合约
- 会依照继承顺序往上找
- 在多重继承时特别重要

## 本章小学生总结

### 一页背完

- Constructor: 出生只跑一次
- contract B is A: B 继承 A
- 父函数要被改: virtual
- 子函数改写: override
- 多重继承: 右边赢

教学提示：继承顺序与 override 写错，是 Solidity 新手最常犯的 bug 来源之一。