# Substrate上的智能合约

parity

# Maggie Dong

Software developer @ParityTech

Background:

- NFT protocol
- Dev Advocate
- Ethereum DApp Development
- Substrate Development

parity

# 什么是智能合约

部署并运行在区块链之上的、”无法“更改逻辑的程序

# Substrate上的contract模块

Contract模块为Substrate runtime提供了 **部署** 和 **运行** WebAssembly合约的方法

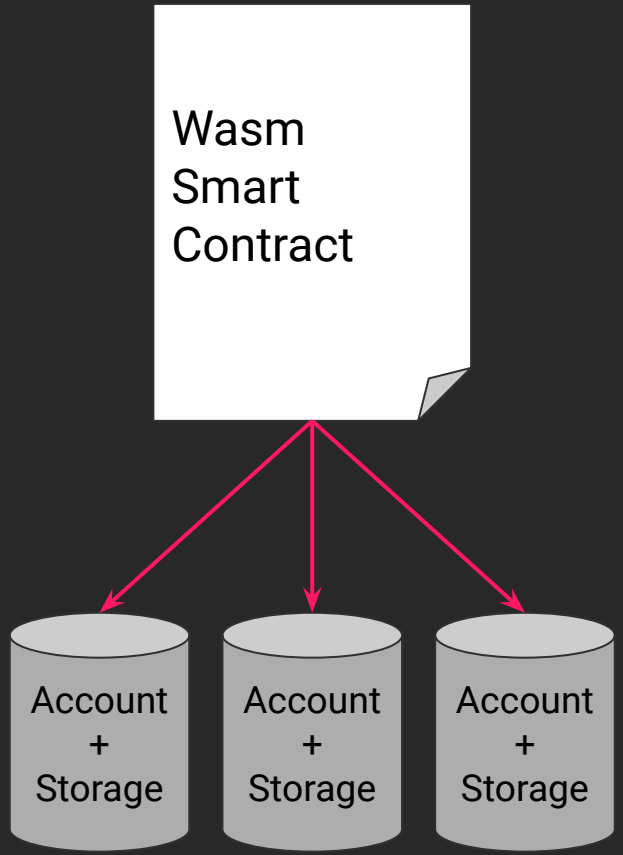
- 基于账户模型
  - 余额
  - 合约代码
  - 存储
- Gas Fees
- 可回滚的交易
- Storage Rent
- Substrate 类型

# 合约部署

## 两步

1. 把Code放到链上 **put_code**
2. Code 实例化 **instantiate**

**可以减少不必要的冗余代码**



# 调用合约

**当调用合约时，会去获取相关代码并且执行**

**调用合约可以：**

- 修改合约账户下的存储
- 创建新的合约
- 调用其他合约
- 调用runtime方法

# 隔离的执行环境

**当调用合约时, 会创造一个新的执行环境:**

- 创建一个wasm实例
- 它有短暂/独立的内存
- 执行环境是被隔离的
- 如果失败了可以回滚状态
- 当全部执行成功后，才去修改substrate的存储

# Parity ink!

用Rust的方式来构建智能合约

parity

# Ink! 的组成


Image by Freepik @ Flaticon



parity

# nk!的结构

| | |
|---|---|
| Language | 真正的eDSL层，为用户提供写代码的语法/方式 |
| Model | 中间层 |
| Core | 和contract模块交互的核心功能层 |

parity

# ink!如何和runtime交互

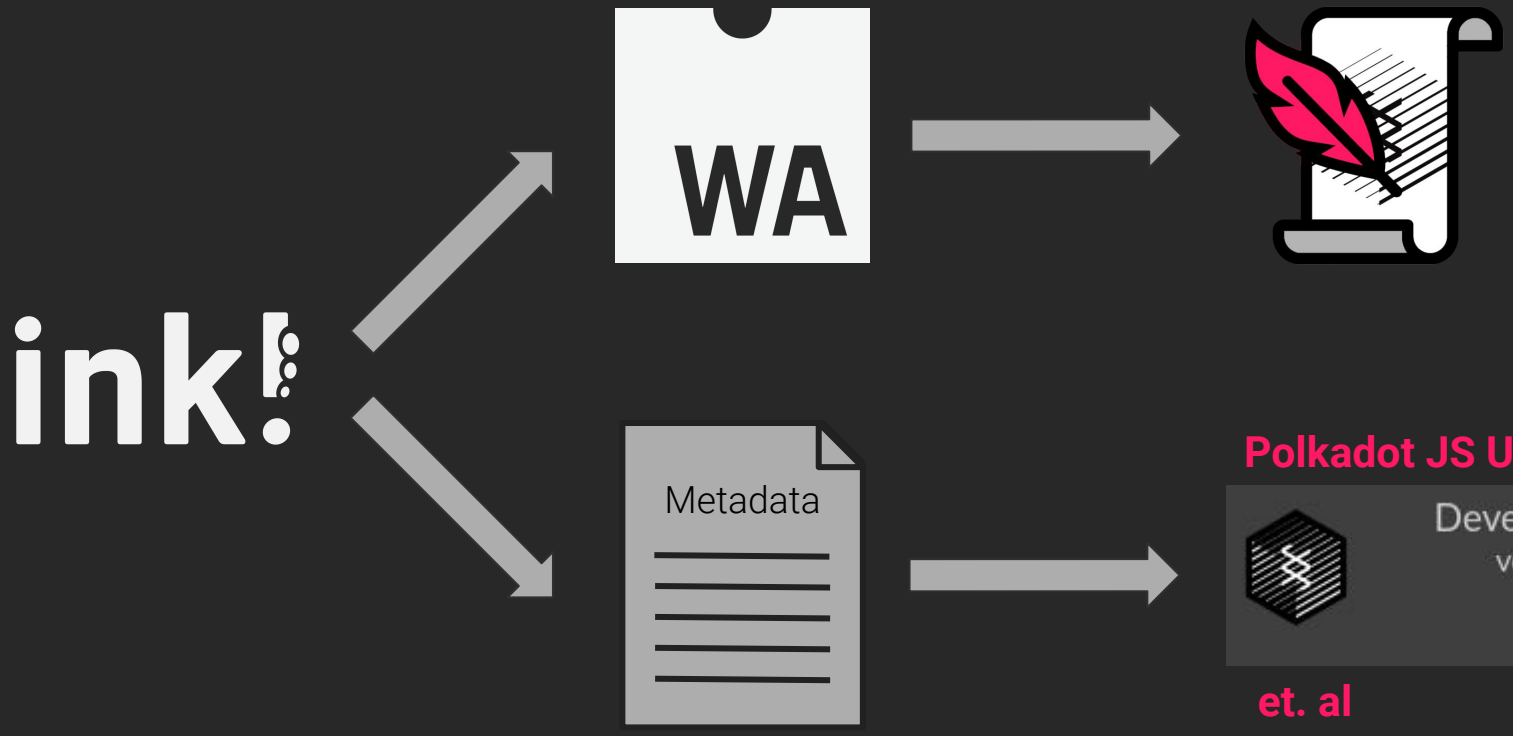
ink!

```
ext_address

ext_caller

ext_call

ext_create

...
```

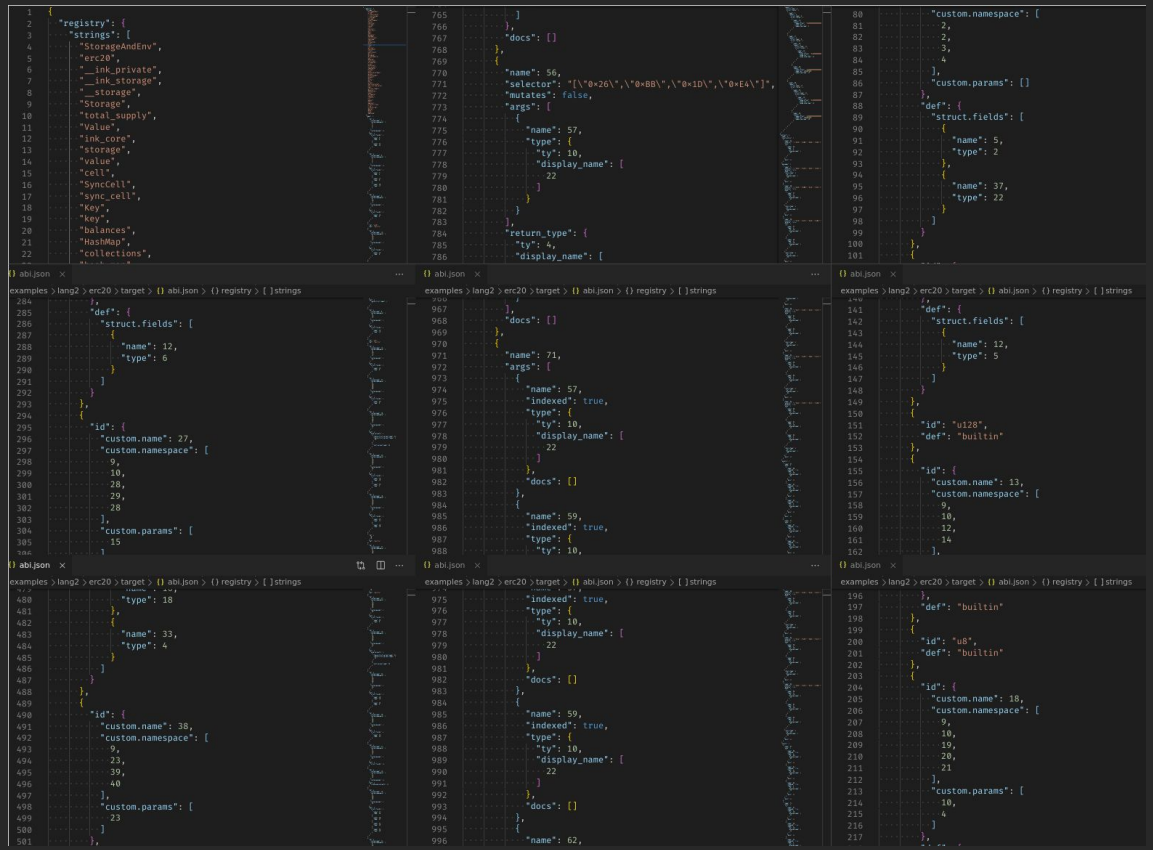


parity

# ink! metadata



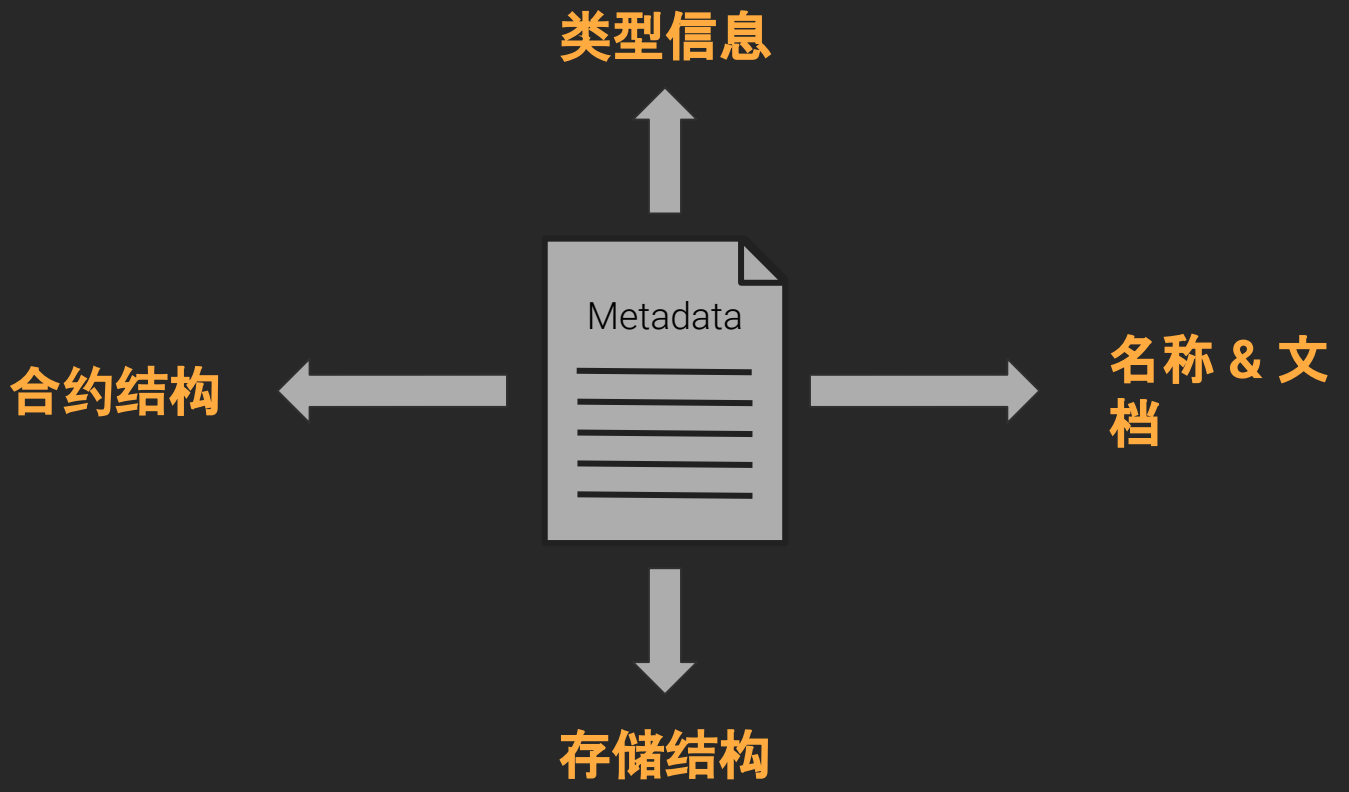
# ink! metadata






parity

# ink! metadata



# Ink! 语法的发展轨迹

```
contract! {
    #![env = DefaultSrmlTypes]

    struct Flipper {
        value: storage::Value<bool>,
    }

    impl Deploy for Flipper {
        fn deploy(&mut self) {
            self.value.set(false)
        }
    }

    impl Flipper {
        pub(external) fn flip(&mut self) {
            *self.value = !*self.value;
        }
    }
}
```

```
#[ink::contract(version = "0.1.0")]
mod flipper {
    #[ink(storage)]
    struct Flipper {
        value: storage::Value<bool>,
    }

    impl Flipper {
        #[ink(constructor)]
        fn new(&mut self) {
            self.value.set(false)
        }

        #[ink(message)]
        fn flip(&mut self) {
            *self.value = !*self.value;
        }
    }
}
```

# 曾经的ink! 语法

```
contract! {
    #![env = DefaultSrmlTypes]

    ink!

}
```



parity

# 现在的ink! 语法

```
#[ink::contract(version = "0.1.0")]
mod flipper {
```

ink! + 

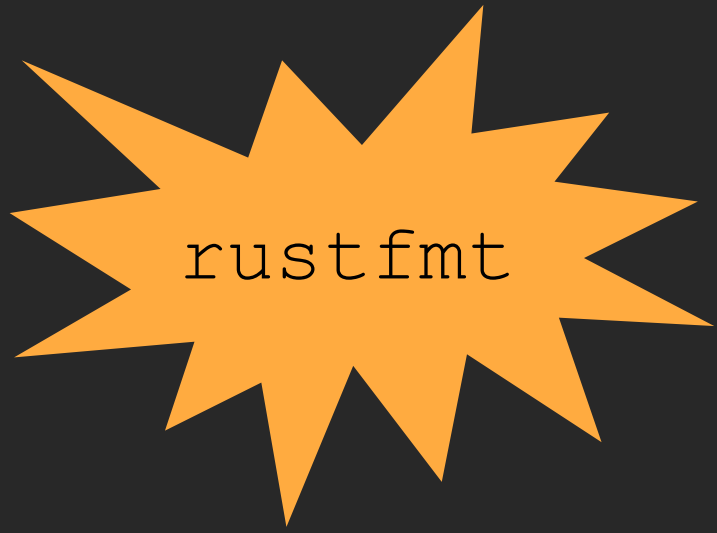
```
}
```

# 修改的好处 - Tooling

```
#[ink::contract(version = "0.1.0")]
mod flipper {
    #[ink(storage)]
    struct Flipper {
value: storage::Value<bool>,
    }

    impl Flipper {
    #[ink(constructor)] fn new(&mut
self) {
            self.value.set(false)
}

        #[ink(message)]
fn flip(&mut self) {
            *self.value
        = !*self.value; }
}
}
```

# 修改的好处 - Tooling

```
#[ink::contract(version = "0.1.0")]
mod flipper {
    #[ink(storage)]
    struct Flipper {
        value: storage::Value<bool>,
    }

    impl Flipper {
        #[ink(constructor)]
        fn new(&mut self) {
            self.value.set(false)
        }

        #[ink(message)]
        fn flip(&mut self) {
            *self.value = !*self.value;
        }
    }
}
```



# 如何写ink!

```
#![cfg_attr(not(feature = "std"), no_std)]
use ink_lang as ink;

#[ink::contract(version = "0.1.0")]
mod mycontract {
    use ink_core::storage;

    #[ink(storage)]
    struct MyContract {
        ...
    }

    #[ink(event)]
    struct MyContract {
    }
    impl MyContract {
        #[ink(constructor)]
        fn new(&mut self, initial_supply: Balance) {
        }
        #[ink(message)]
        fn call_a(&self) -> Balance {
        }

        #[ink(message)]
        fn call_b(&self, owner: AccountId) -> Balance {
        }
```

parity

# ink! Header

```
#[ink::contract(
    version = "0.1.0",
    env = DefaultSrmlTypes,
)]
mod contract {
    …
}
```

# ink! Header

```
#[ink::contract(
    version = "0.1.0",
    env = DefaultSrmlTypes,
)]
mod contract {
    …
}
```

# ink! storage


```
#[ink(storage)]
struct Erc20 {
    total_supply: storage::Value<Balance>,
    balances: storage::HashMap<AccountId, Balance>,
    allowances: storage::HashMap<
            (AccountId, AccountId), // Key
            Balance                // Value
    >,
}
```

# ink! events

```
#[ink(event)]
struct Transfer {
    #[ink(topic)] from: Option<AccountId>,
    #[ink(topic)] to: Option<AccountId>,
    value: Balance,
}
```

定义

USAGE

```
self.env().emit_event(Transfer {
    from: Some(from),
    to: Some(to),
    value,
});
```

# ink! constructors

```
#[ink(constructor)]
fn new(&mut self, initial_supply: Balance) {
    let caller = self.env().caller();
    self.total_supply.set(initial_supply);
    self.balances.insert(caller, initial_supply);
}
```

# ink! messages

```
#[ink(message)]
fn total_supply(&self) -> Balance {
    self.total_supply.get()
}
```

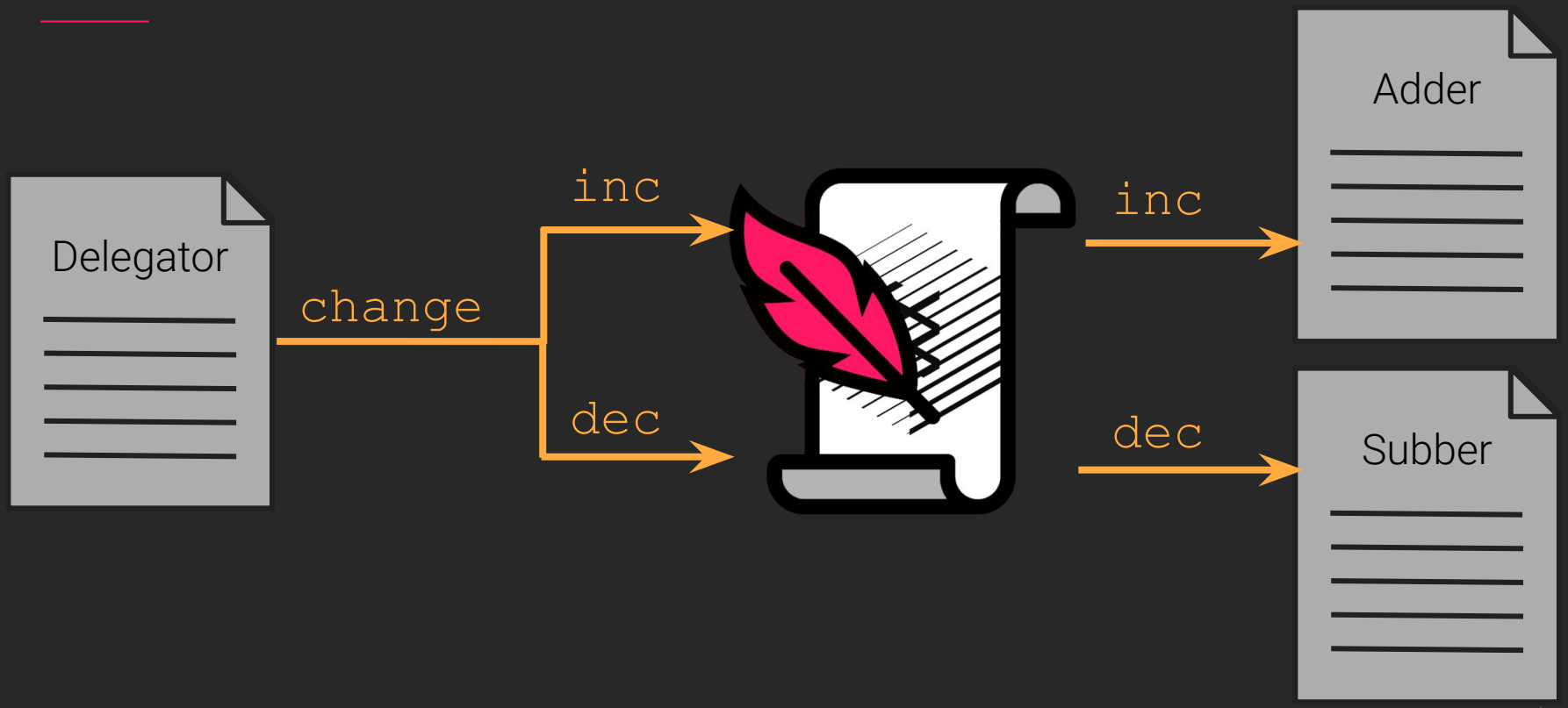
# ink! errors

```
error: #[ink(constructor)] functions must have a &mut self receiver
  --> src/lib.rs:52:16
   |
52 |         fn new(&self, initial_supply: Balance) {
   |                ^^^^^


error: #[ink(constructor)] functions must not have a return type
  --> src/lib.rs:52:52
   |
52 |         fn new(&mut self, initial_supply: Balance) -> Self {
   |                                                    ^^^^^^^
```

# ink! 跨合约调用

```
/// Delegates to either Adder or Subber contract.
#[ink(message)]
fn change(&mut self, by: i32) {
    match self.which {
        Which::Adder => self.adder.inc(by),
        Which::Subber => self.subber.dec(by),
    }
}
```

# ink! across borders



# Ink! 支持的类型

## 存储类型

AccountId
Balance
Hash
Timestamp
Balance
BlockNumber
Call

# Ink! 支持的类型

## 基本存储类型

Bitstash
Bitvec
Hashmap
Smallvec
Stash - **可索引的元素**
Vec
Lazy/Value - **单值**

## 复合存储类型

Struct
enum

# Ink! 环境安装

**Substrate的要求**

```
curl https://getsubstrate.io -sSf | bash -s -- --fast
rustup target add wasm32-unknown-unknown --toolchain stable
rustup component add rust-src --toolchain nightly
```

**安装指定版本Substrate**

```
cargo install node-cli --git
https://github.com/paritytech/substrate.git --tag
v2.0.0-rc4 --force
```

**安装cargo contract插件**

```
cargo install cargo-contract --vers 0.6.1 --force
```

# 如何创建ink! 项目

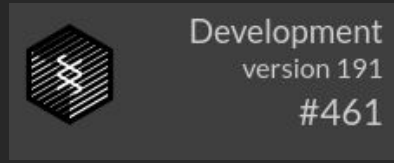
cargo contract new <your-project>

# 如何测试ink! 项目

cargo +nightly test

# Ink! 交互

Polkadot JS UI



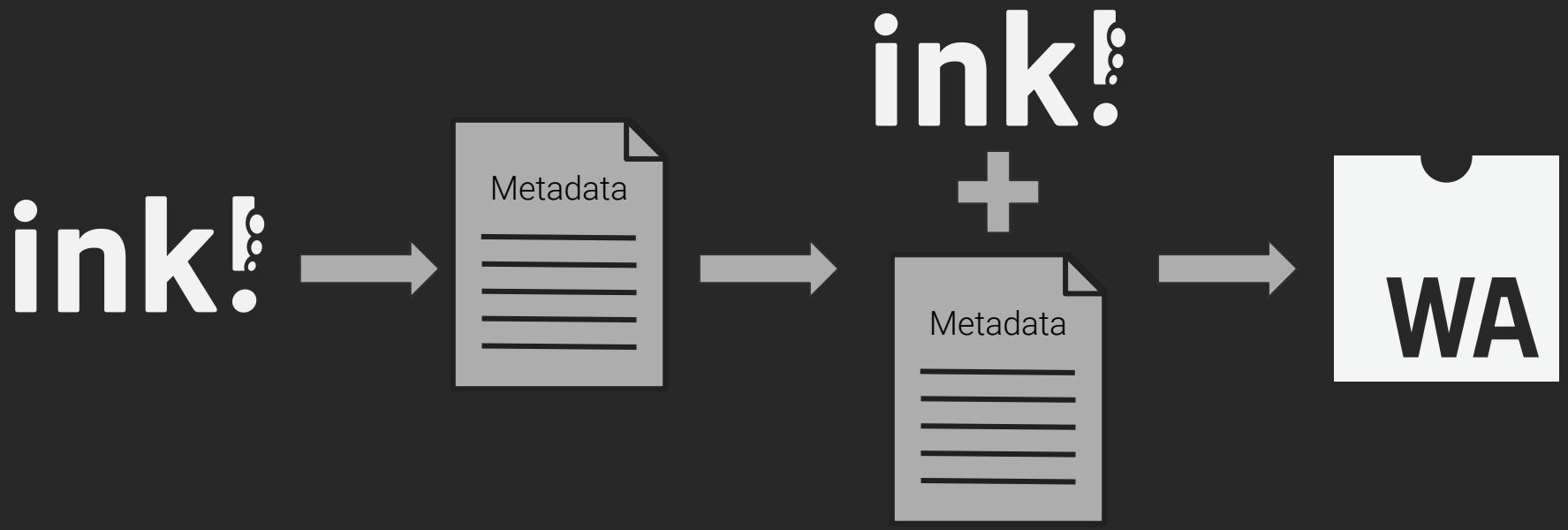

**演示**



parity

# future<ink!>

# Bake it; Eat it



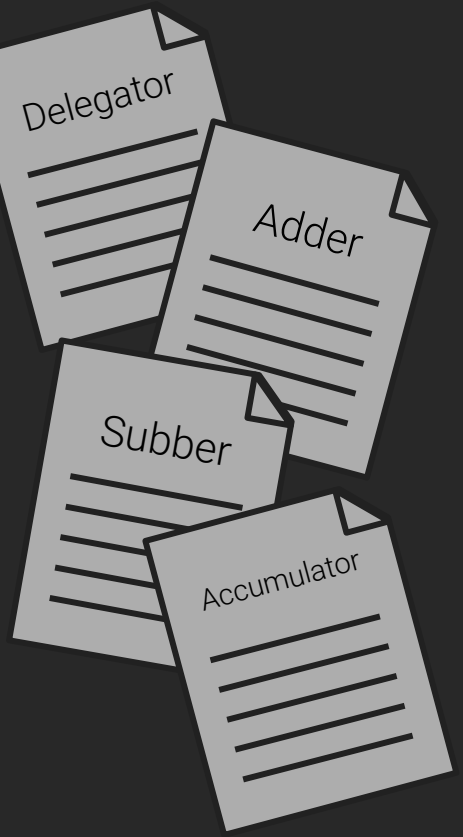
# Trait & Generics Support

```
#[ink::contract(version = "0.1.0")]
mod callback {
    #[ink(trait)]
    trait Listen {
        fn listen(&self, value: i32);
    }

    #[ink(storage)]
    struct Callback<T> {
        listener: storage::Value<T>,
    }

    …
}
```

```
impl<T> Callback<T>
where
    T: Listen,
{
    #[ink(constructor)]
    fn new(&mut self, listener: T) {
        self.listener.set(listener)
    }

    #[ink(message)]
    fn fire(&self, value: i32) {
        self.listener.call(value)
    }
}
```

# Multi-file Projects

```
#[ink::contract(version = "0.1.0")]
mod delegator {
    #[ink(storage)]
    struct Delegator { ... }

    #[ink(path = "accumulator")] mod accumulator;
    #[ink(path = "adder")] mod adder;
    #[ink(path = "subber")] mod subber;

    impl Delegator {
        #[ink(constructor)]
        fn new(&mut self, ...) { ... }
        #[ink(message)]
        fn delegate(&self, value: i32) { ... }
    }
}
```





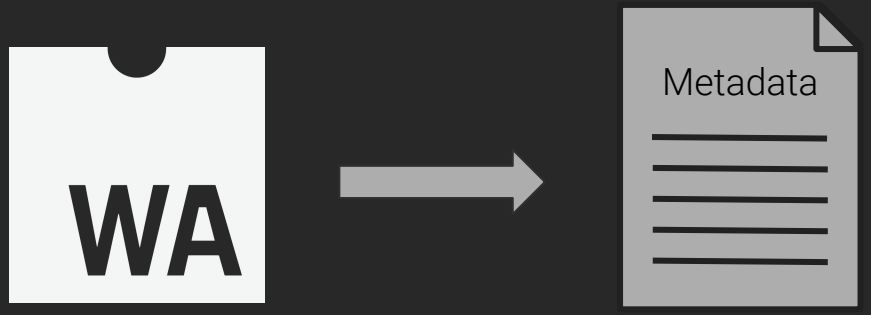

parity

# Pre- & Post Conditions

```
fn transfer(&mut self, from: AccountId, to: AccountId, value: Balance) {
    let from_balance = .. ;
    let to_balance = .. ;
    assume!(
        self.approval(from, to) >= value &&
        from_balance >= value
    );
    // do the computations that rely on the assumptions
    ensure!(
        self.approval(from, to) >= 0 &&
        self.balance_of(from) = from_balance - value &&
        self.balance_of(to) = to_balance + value
    );
}
```

# Built-ink! Errors

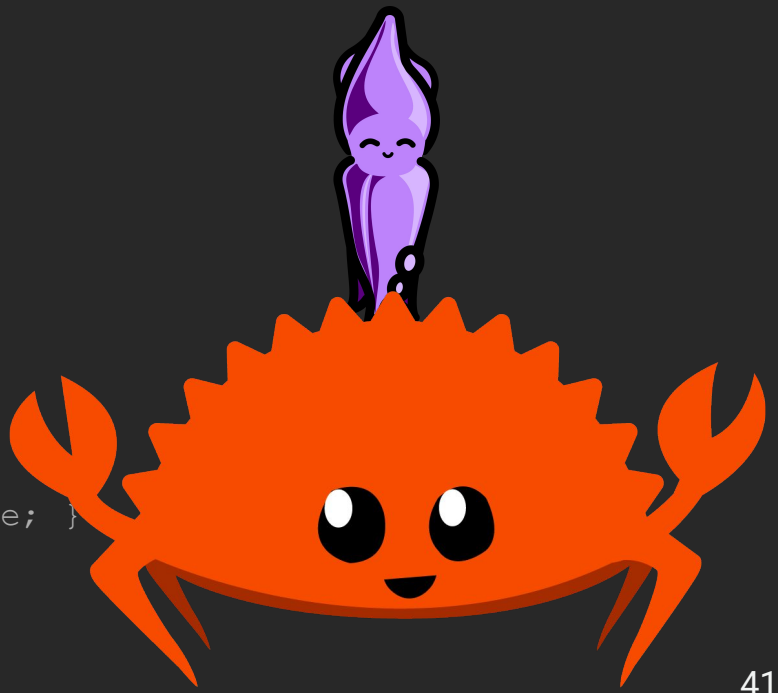
```rust
#[ink(error)]
enum Error {
    /// Invalid message input.
    InvalidInput,
    /// Too few elements to process.
    TooFewElements,
    /// The value of the user is insufficient.
    InsufficientValue,
}
```



# Rust Crate Attributes

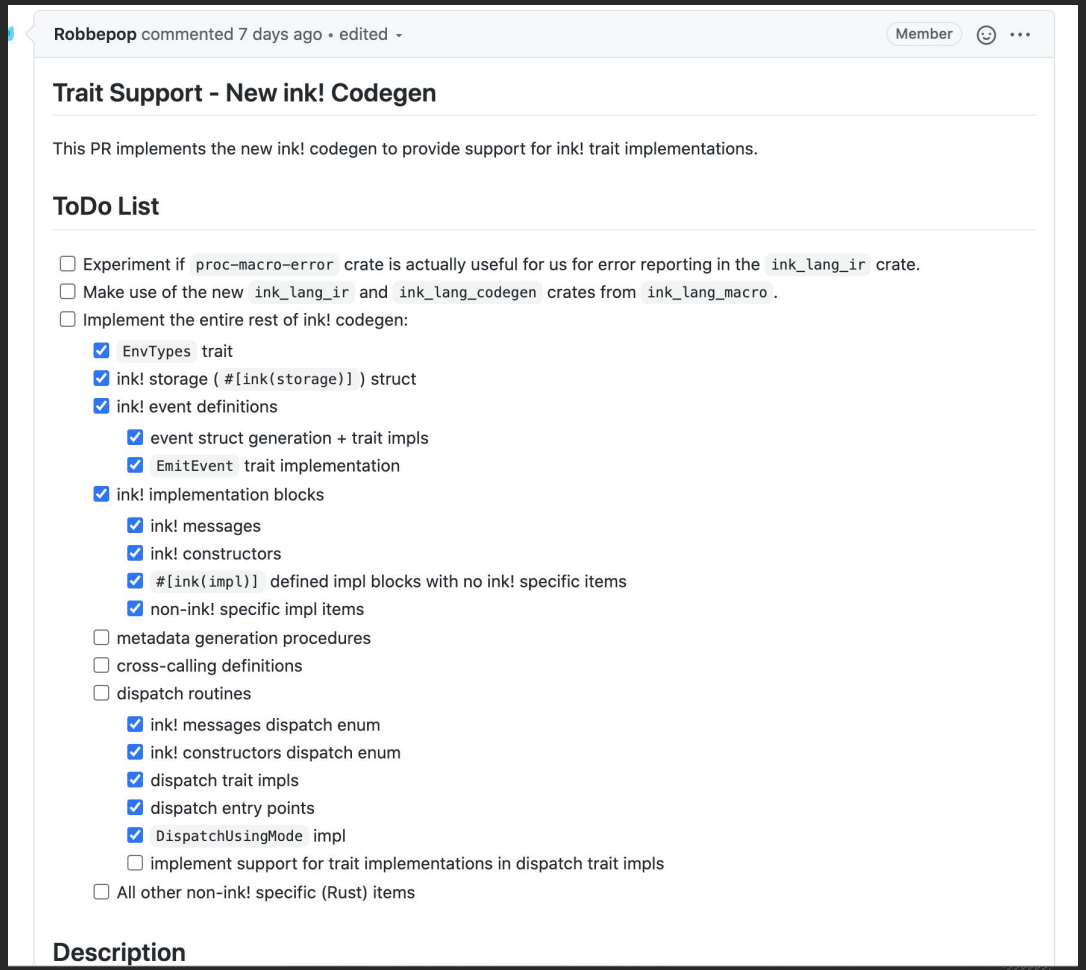
crate root

```
#![ink::contract(version = "0.1.0")]

#[ink(storage)]
struct Flipper {
    value: storage::Value<bool>,
}

impl Flipper {
    #[ink(constructor)]
    fn new(&mut self) { self.value.set(false) }

    #[ink(message)]
    fn flip(&mut self) { *self.value = !*self.value; }

    #[ink(message)]
    fn get(&self) -> bool { *self.value }
}
```



# 即将发布的ink!3.0

https://github.com/paritytech/ink/pull/470



**Robbepop** commented 7 days ago • edited — Member

## Trait Support - New ink! Codegen

This PR implements the new ink! codegen to provide support for ink! trait implementations.

## ToDo List

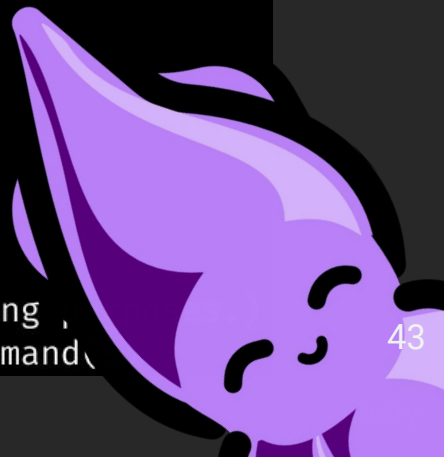
- [ ] Experiment if `proc-macro-error` crate is actually useful for us for error reporting in the `ink_lang_ir` crate.
- [ ] Make use of the new `ink_lang_ir` and `ink_lang_codegen` crates from `ink_lang_macro`.
- [ ] Implement the entire rest of ink! codegen:
  - [x] `EnvTypes` trait
  - [x] ink! storage ( `#[ink(storage)]` ) struct
  - [x] ink! event definitions
    - [x] event struct generation + trait impls
    - [x] `EmitEvent` trait implementation
  - [x] ink! implementation blocks
    - [x] ink! messages
    - [x] ink! constructors
    - [x] `#[ink(impl)]` defined impl blocks with no ink! specific items
    - [x] non-ink! specific impl items
  - [ ] metadata generation procedures
  - [ ] cross-calling definitions
  - [ ] dispatch routines
    - [x] ink! messages dispatch enum
    - [x] ink! constructors dispatch enum
    - [x] dispatch trait impls
    - [x] dispatch entry points
    - [x] `DispatchUsingMode` impl
    - [ ] implement support for trait implementations in dispatch trait impls
  - [ ] All other non-ink! specific (Rust) items

## Description

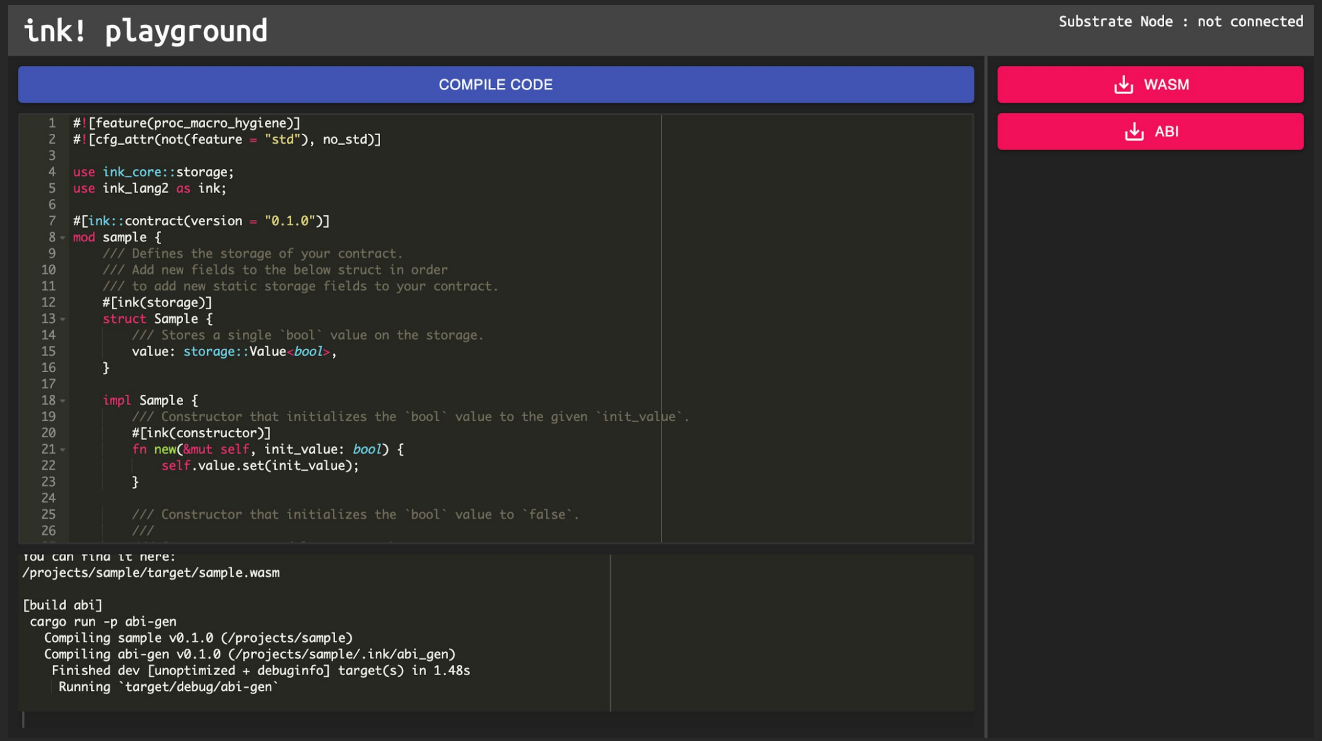
parity

# ink! CLI

```
cargo-contract 0.1.2
Parity Technologies <admin@parity.io>
Utilities to develop Wasm smart contracts.

USAGE:
    cargo contract <SUBCOMMAND>

OPTIONS:
    -h, --help       Prints help information
    -V, --version    Prints version information

SUBCOMMANDS:
    new       Setup and create a new smart contract.
    build     Builds the smart contract.
    test      Test the smart contract off-chain.
    deploy    Deploy the smart contract on-chain. (Also for testing )
    help      Prints this message or the help of the given subcommand(
```



# ink! playground




ink-playground.com



parity