CESS

Cumulus Encrypted Storage System

CESS Course Week 2 - Episode 4

dAapp Development

Ink! Smart Contract

CESS Official Website

Table of Contents

CESS
Cumulus Encrypted Storage System

❖ Overview of Types of Development on CESS	Difficulty
➢ Interacting with CESS using DeOSS API and SDKs	★★
➢ Building and Deploying Smart Contracts	★★★
➢ Building Custom Pallets to Integrate with CESS	★★★★
❖ dApp Development using Ink! smart contract	
❖ Interacting with Ink! smart contracts (use-ink)	
❖ NFT Marketplace example	

CESS

Cumulus Encrypted Storage System

dApp Development using Ink!

What is Ink!?

- Rust-based Language
- Compiles to WebAssembly (Wasm)
- Support for Complex Logic
- Runtime Environment

CESS

Cumulus Encrypted Storage System

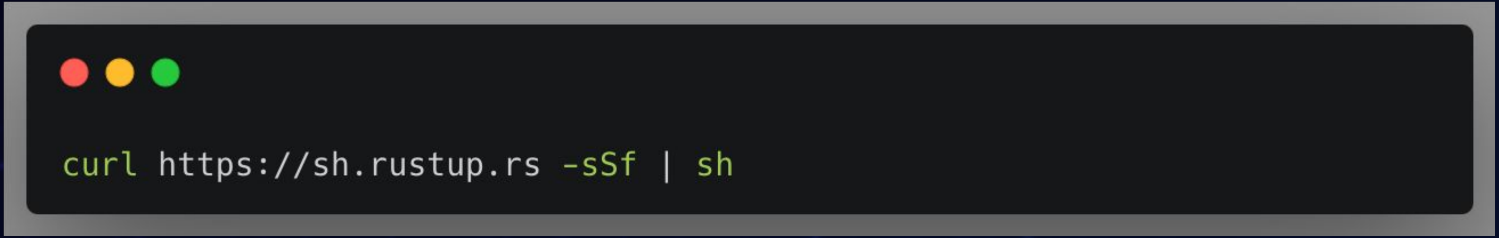
Ink! Smart Contract

Flipper

Prerequisites

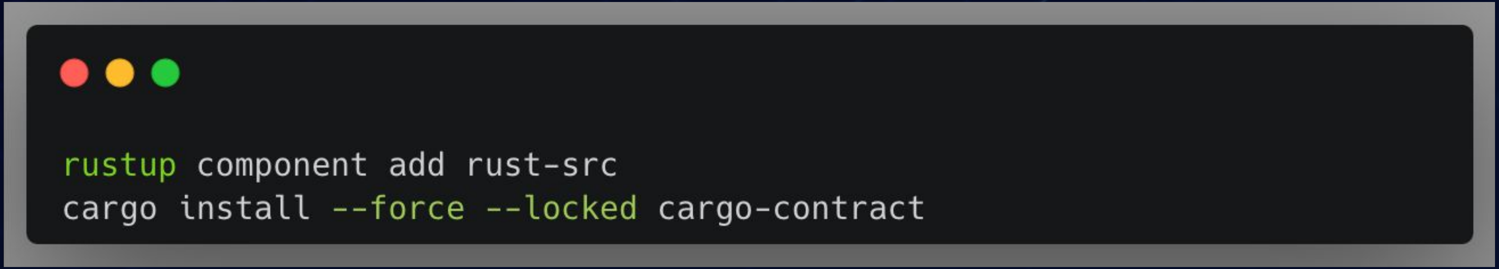
CESS
Cumulus Encrypted Storage System

❖ Rust & Cargo



```
curl https://sh.rustup.rs -sSf | sh
```

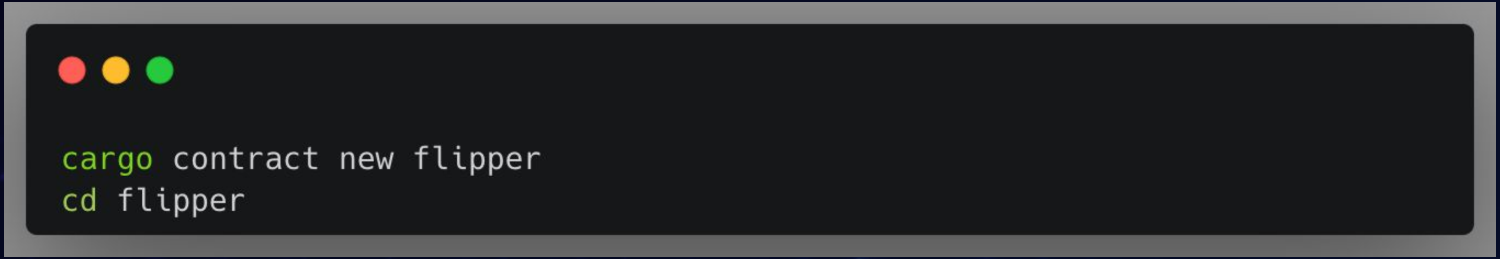
❖ Ink! CLI



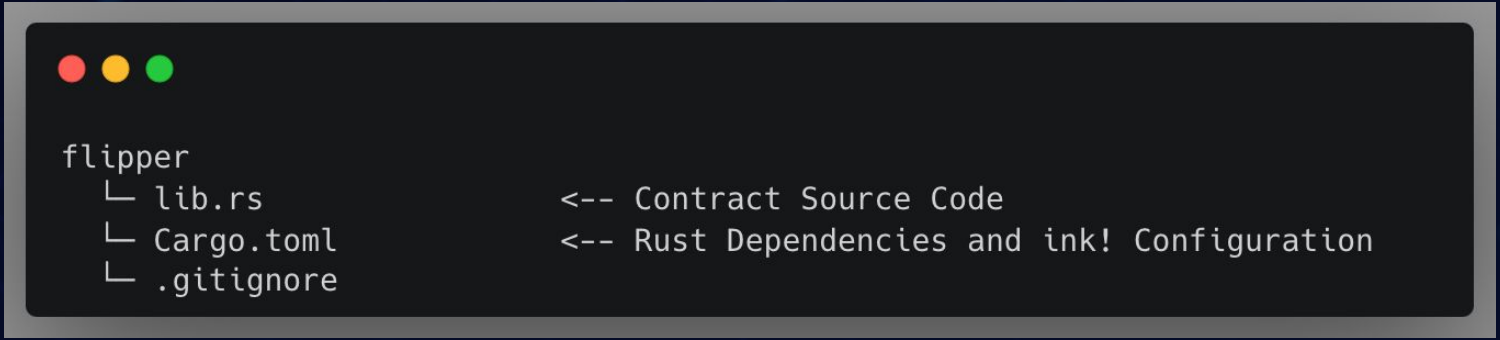
```
rustup component add rust-src
cargo install --force --locked cargo-contract
```

Walkthrough a simple ink! Smart Contract

Flipper Project



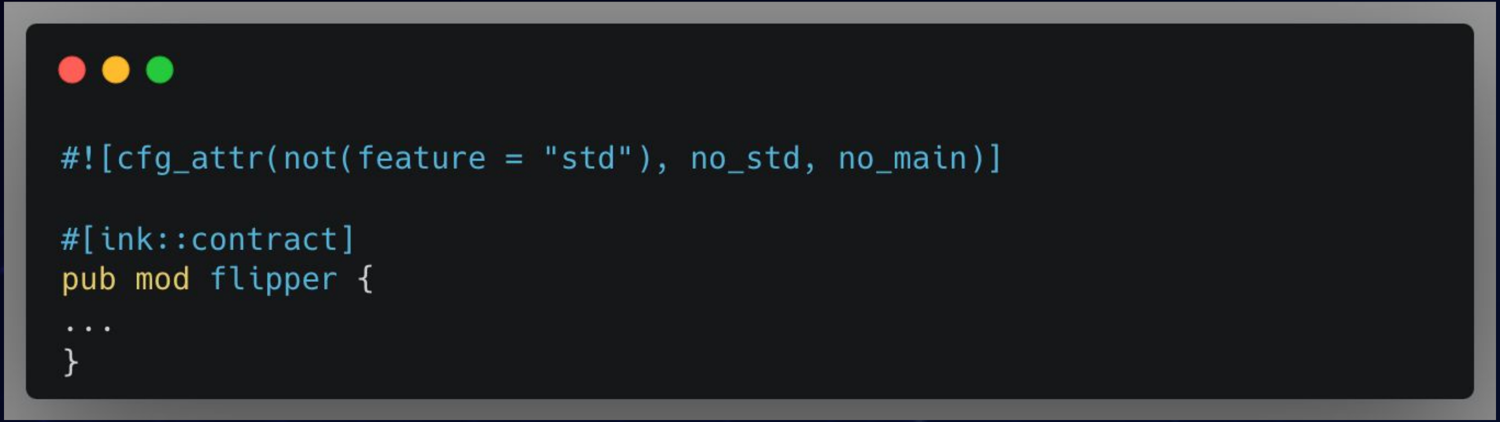
```
cargo contract new flipper
cd flipper
```



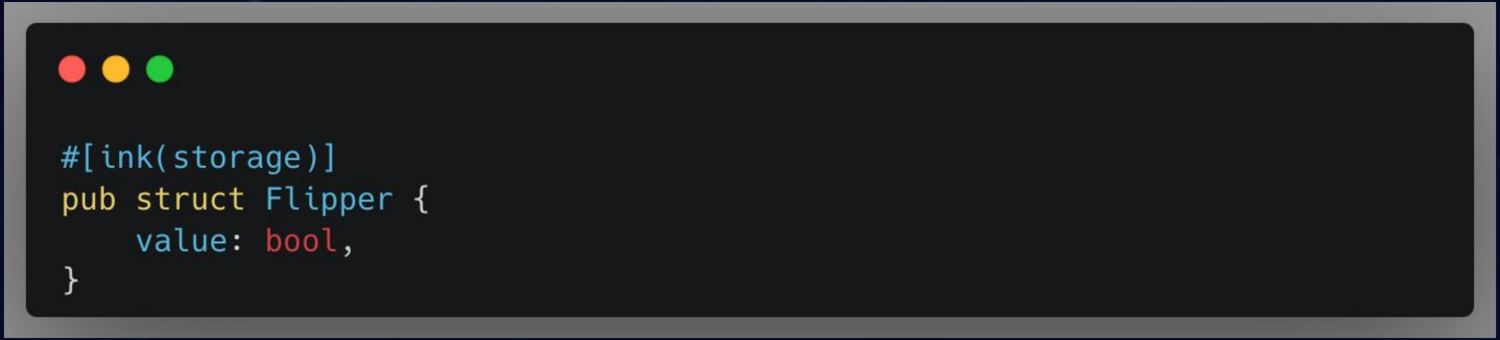
```
flipper
  └ lib.rs                   <-- Contract Source Code
  └ Cargo.toml               <-- Rust Dependencies and ink! Configuration
  └ .gitignore
```

Lib.rs Structure

CESS
Cumulus Encrypted Storage System

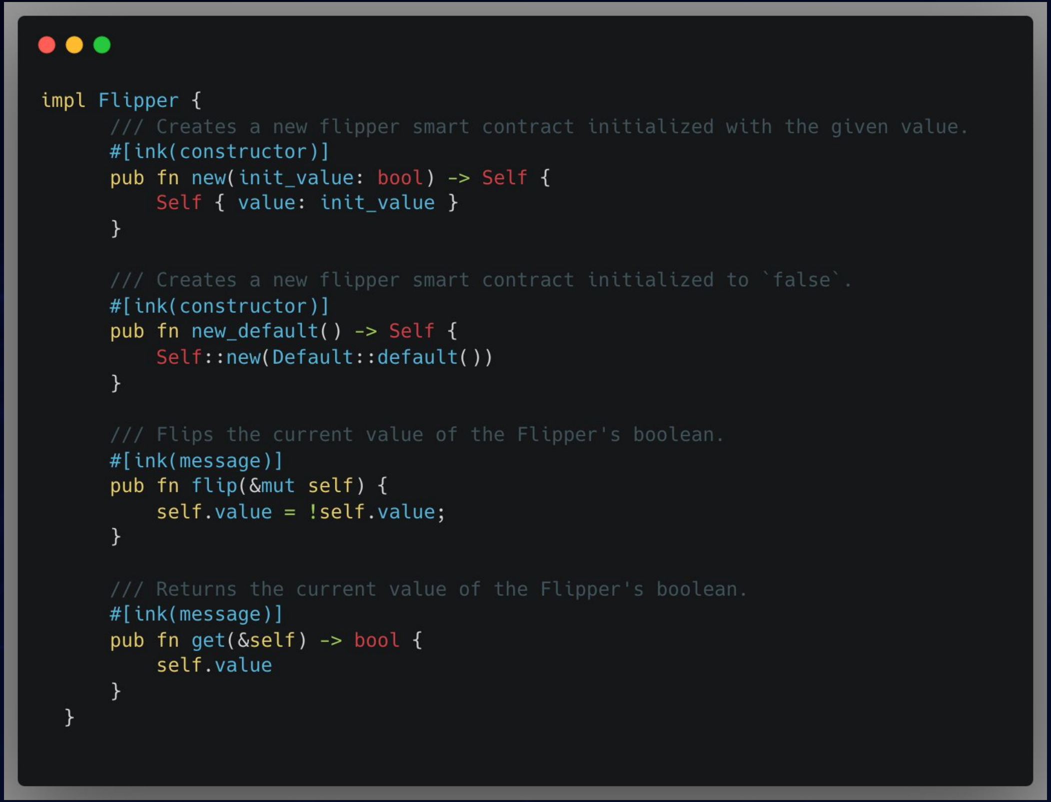


```
#![cfg_attr(not(feature = "std"), no_std, no_main)]

#[ink::contract]
pub mod flipper {
...
}
```



```
#[ink(storage)]
pub struct Flipper {
    value: bool,
}
```

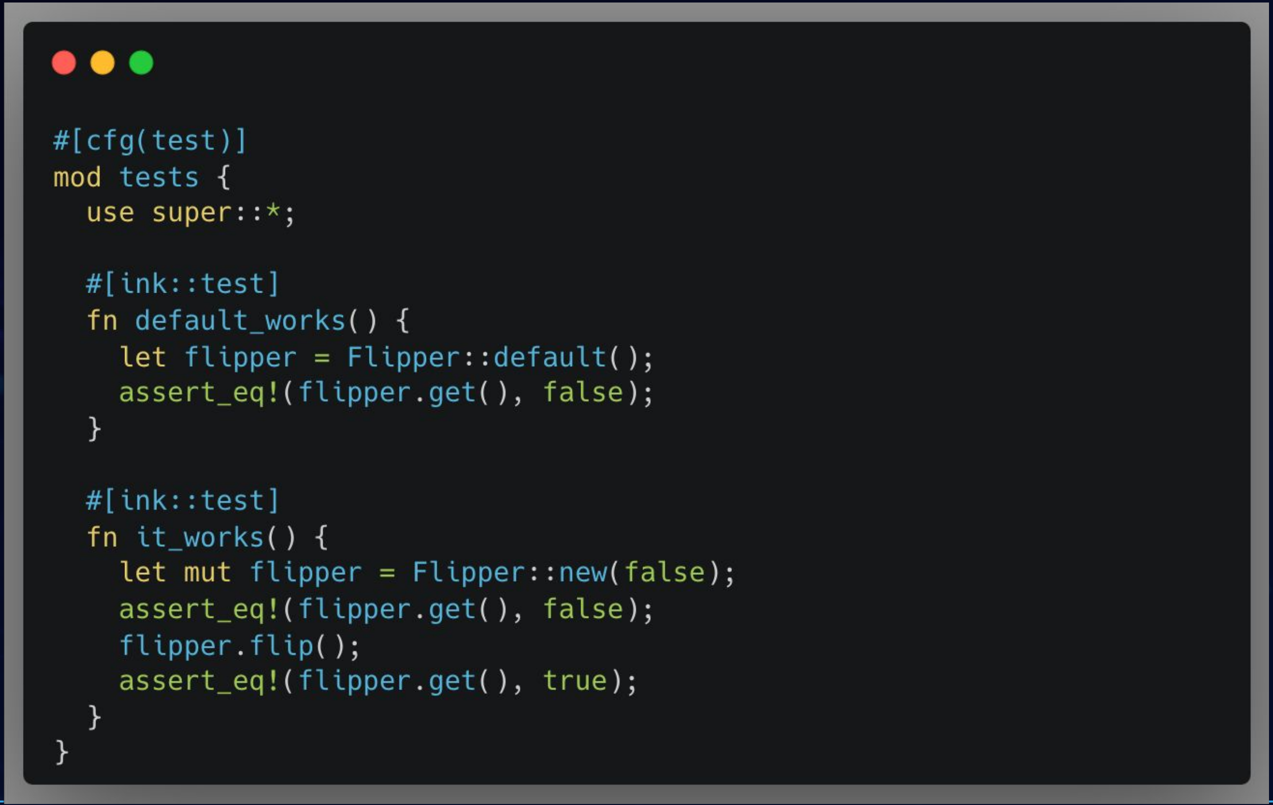
Flipper Implementation



```
impl Flipper {
      /// Creates a new flipper smart contract initialized with the given value.
      #[ink(constructor)]
      pub fn new(init_value: bool) -> Self {
          Self { value: init_value }
      }

      /// Creates a new flipper smart contract initialized to `false`.
      #[ink(constructor)]
      pub fn new_default() -> Self {
          Self::new(Default::default())
      }

      /// Flips the current value of the Flipper's boolean.
      #[ink(message)]
      pub fn flip(&mut self) {
          self.value = !self.value;
      }

      /// Returns the current value of the Flipper's boolean.
      #[ink(message)]
      pub fn get(&self) -> bool {
          self.value
      }
  }
```

Test Module

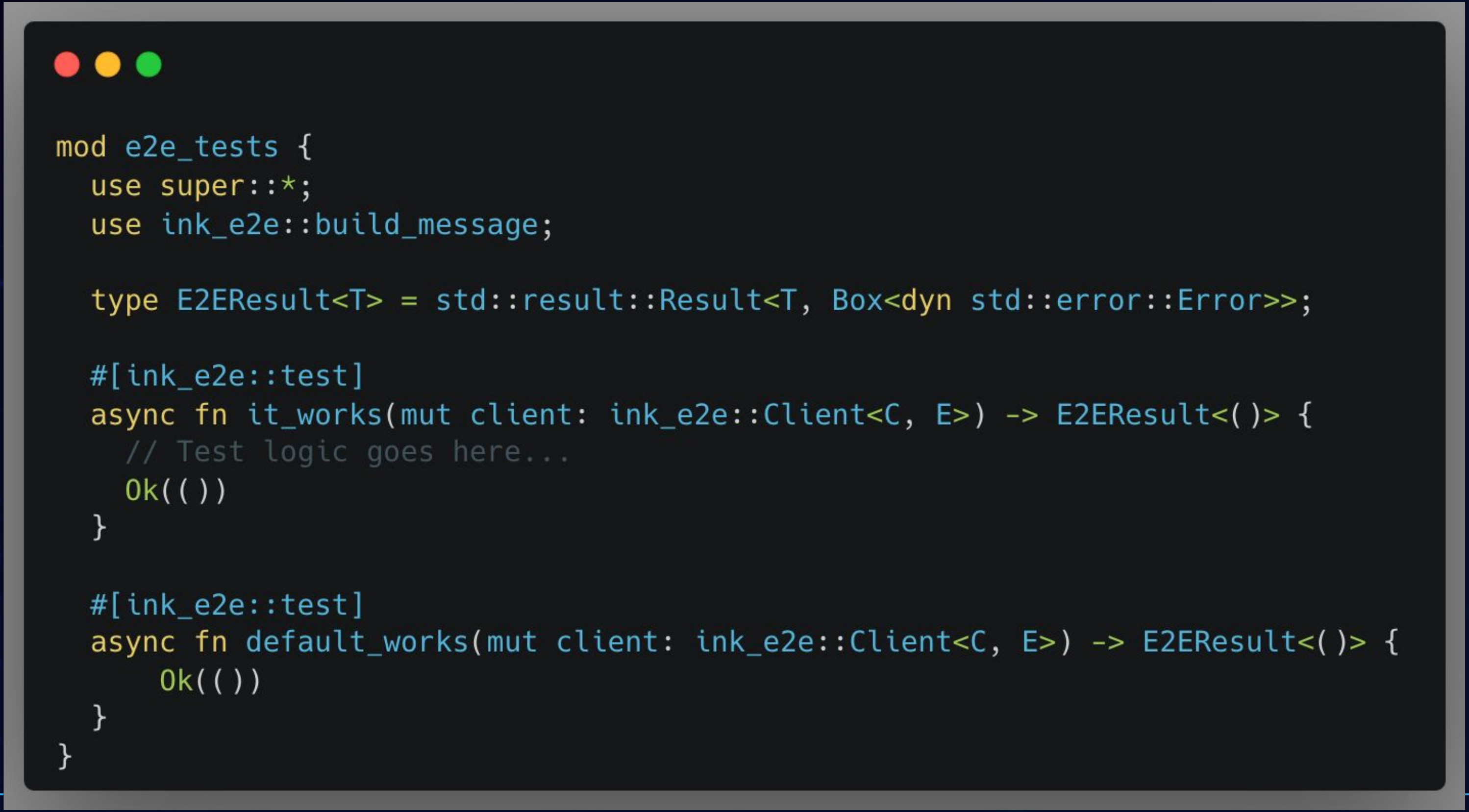
CESS
Cumulus Encrypted Storage System



```
#[cfg(test)]
mod tests {
  use super::*;

  #[ink::test]
  fn default_works() {
    let flipper = Flipper::default();
    assert_eq!(flipper.get(), false);
  }

  #[ink::test]
  fn it_works() {
    let mut flipper = Flipper::new(false);
    assert_eq!(flipper.get(), false);
    flipper.flip();
    assert_eq!(flipper.get(), true);
  }
}
```

E2E(End to End) Test

CESS
Cumulus Encrypted Storage System



```
mod e2e_tests {
  use super::*;
  use ink_e2e::build_message;

  type E2EResult<T> = std::result::Result<T, Box<dyn std::error::Error>>;

  #[ink_e2e::test]
  async fn it_works(mut client: ink_e2e::Client<C, E>) -> E2EResult<()> {
    // Test logic goes here...
    Ok(())
  }

  #[ink_e2e::test]
  async fn default_works(mut client: ink_e2e::Client<C, E>) -> E2EResult<()> {
      Ok(())
  }
}
```

CESS

Cumulus Encrypted Storage System

Deploy Ink! Smart Contract On CESS

CESS

Cumulus Encrypted Storage System

Interacting with Ink! Smart Contract Using The

useink Library

CESS

Cumulus Encrypted Storage System

NFT Marketplace
Example

Contract Features

CESS

Cumulus Encrypted Storage System

- ★ Minting NFTs
- ★ Listing and Purchasing
- ★ Owner Controls
- ★ Withdrawal of Funds
- ★ Flexible