**Implementing the Receive Fallback**

**Introduction:**

In Solidity, if Ether is sent to a contract without a `receive` or `fallback` function, the transaction will be rejected, and the Ether will not be transferred. In this lesson, we’ll explore how to handle this scenario effectively.

**Receive and fallback functions:**

`receive` and `fallback` are special functions triggered when users send Ether directly to the contract or call non-existent functions. These functions do not return anything and must be declared `external`**.**

To illustrate, let’s create a simple contract:

//SPDX-License-Identifier: MIT

Pragma solidity ^0.8.7;

Contract FallbackExample {

Uint256 public result;

Receive() external payable {

Result = 1;

}

Fallback() external payable {

Result = 2;

}

}

In this contract, `result` is initialized to zero. When Ether is sent to the contract, the `receive` function is triggered, setting `result` to one. If a transaction includes data but the specified function does not exist, the `fallback` function will be triggered, setting `result` to two. For a comprehensive explanation, refer to SolidityByExample (<https://solidity-by-example.org/fallback/>).

// Ether is sent to contract

// is msg.data empty?

// / \

// yes no

// / \

// receive()? Fallback()

// / \

// yes no

// / \

//receive() fallback()

**Sending Ether to fundMe:**

When a user sends Ether directly to the `fundMe` contract without calling the `fund` function, the `receive` function can be used to redirect the transaction to the `fund` function:

Receive() external payable {

Fund();

}

Fallback() external payable {

Fund();

}

To test this functionality, send some Sepolia Ether to the `fundMe` contract using MetaMask. This does not directly call the `fund` function, but the `receive` function will trigger it. After confirming the transaction, you can check the `funders` array to see that it has been updated, reflecting the successful invocation of the `fund` function by the `receive` function.

This approach ensures that all transactions are processed as intended. Although directly calling the `fund` function costs less gas, this method ensures the user’s contribution is properly acknowledged and credited.

**Conclusion:**

By implementing `receive` and `fallback` functions, contracts can handle direct Ether transfers and non-existent function calls effectively, ensuring that transactions are processed as intended and users’ contributions are properly tracked.