**Sending ETH from a Contract**

**Introduction:**

This lesson explores three different methods of sending ETH from one account to another: `transfer`, `send`, and `call`. We will understand their differences, how each one works, and when to use **one** instead of another.

**Transfer:**

The `transfer` function is the simplest way to send Ether to a recipient address:

Payable(msg.sender).transfer(amount); // the current contract sends the Ether amount to the msg.sender

It’s necessary to convert the recipient address to a \*\*payable\*\* address to allow it to receive Ether. This can be done by wrapping `msg.sender` with the `payable` keyword.

However, `transfer` has a significant limitation. It can only use up to 2300 gas and it reverts any transaction that exceeds this gas limit, as illustrated by [Solidity by Example](<https://solidity-by-example.org/sending-ether/>).

**Send:**

The `send` function is similar to `transfer`, but it differs in its behaviour:

Bool success = payable(msg.sender).send(address(this).balance);

Require(success, “Send failed”);

Like `transfer`, `send` also has a gas limit of 2300. If the gas limit is reached, it will not revert the transaction but return a boolean value (`true` or `false`) to indicate the success or failure of the transaction. It is the developer's responsibility to handle failure correctly, and it’s good practice to trigger a \*\*revert\*\* condition if the `send` returns `false`.

**Call:**

The `call` function is flexible and powerful. It can be used to call any function \*\*without requiring its ABI\*\*. It does not have a gas limit, and like `send`, it returns a boolean value instead of reverting like `transfer`.

(bool success, ) = payable(msg.sender).call{value: address(this).balance}(“”);

Require(success, “Call failed”);

To send funds using the `call` function, we convert the address of the receiver to `payable` and add the value inside curly brackets before the parameters passed.

The `call` function returns two variables: a boolean for success or failure, and a byte object which stores returned data if any

**Important:** `call` is the recommended way of sending and receiving Ethereum or other blockchain native tokens.

**Conclusion:**

In conclusion, transfer, send, and call are three unique methods for transferring Ether in Solidity. They vary in their syntax, behaviour, and gas limits, each offering distinct advantages and drawbacks.