Fallback() and Receive() Function

In Solidity, `fallback` and `receive` are special functions that handle incoming calls to a contract, particularly when Ether is sent, or a non-existent function is called.

1. `receive()` Function:

- **Purpose:** This function is triggered **when the contract receives plain Ether transfers** (without any accompanying calldata).
- **Declaration:**

```
receive() external payable {
    // This runs when someone sends ETH directly to the contract
}
```

- **Conditions for Execution:**
  - Ether is sent to the contract.
  - The `msg.data` (calldata) is empty.
- **Characteristics:**
  - Must be declared `external` and `payable`.
  - Cannot have arguments.
  - Cannot return any value.
  - More gas-efficient than `fallback()` for simple Ether transfers.
- **Example:**

```
contract Test {
    event Received(address sender, uint amount);

    receive() external payable {
        emit Received(msg.sender, msg.value);
    }
}
```

2. `fallback()` Function:

- **Purpose:** A more general-purpose function that is executed when no other function matches the call, or when Ether is sent with non-empty calldata.
- **Declaration:**

```
fallback() external payable {
    // Handle calls to non-existent functions or Ether with calldata
}
```

- **Conditions for Execution:**
    - A function is called that does not exist in the contract.
    - Ether is sent to the contract, and `msg.data` is not empty.
    - Ether is sent to the contract, `msg.data` is empty, but no `receive()` function is defined.
- **Characteristics:**
    - Must be declared `external` and `payable` (if it should receive Ether).
    - Can optionally take `bytes` as input to access the `msg.data`.
    - Can be used to implement custom logic for handling unexpected calls.
    - Has a 2300 gas limit when called by `transfer()` or `send()`, limiting the complexity of operations within it.
- **Example:**

```
contract Test {
    event FallbackCalled(address sender, uint amount, bytes data);

    fallback() external payable {
        emit FallbackCalled(msg.sender, msg.value, msg.data);
    }
}
```

Key Differences:

- **Calldata:** `receive()` is triggered only when `msg.data` is empty, while `fallback()` is triggered when `msg.data` is not empty, or when no other function matches and `receive()` is absent.
- **Purpose:** `receive()` is specifically for receiving plain Ether, whereas `fallback()` handles both Ether with calldata and calls to non-existent functions.
- **Precedence:** If both `receive()` and `fallback()` are present, and plain Ether is sent (empty calldata), `receive()` will be executed. `fallback()` will only be executed if `receive()` is absent or if `msg.data` is not empty.

Complete Example:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract EtherHandler {
    event Received(uint amount);
    event FallbackCalled(address sender, uint amount, bytes data);

    // Called when no data is sent
    receive() external payable {
        emit Received(msg.value);
    }

    // Called when data is sent or wrong function is called
    fallback() external payable {
        emit FallbackCalled(msg.sender, msg.value, msg.data);
    }
}
```