## **🔹 What are Access Modifiers?**

Access modifiers in Solidity control **who can access functions and variables** in a smart contract. This helps **protect sensitive data** and **prevent unauthorized actions**.

### **🔹 Types of Access Modifiers & When to Use Them**

|  |  |  |
| --- | --- | --- |
| **Modifier** | **Who Can Access?** | **When to Use?** |
| public | **Anyone** (external users & other contracts) | For functions or variables that should be visible to everyone (e.g., getBalance()) |
| private | **Only inside the contract** | For internal data that should **never** be accessed outside (e.g., sensitive mappings like invoiceExists) |
| internal | **Only inside the contract & derived contracts** | When a variable or function should be accessible within the contract **and its child contracts** |
| external | **Only external accounts or contracts** | For functions that should only be called **from outside** (not from inside the contract) |

## **🔹 Key Points to Consider When Choosing Access Modifiers**

### **✅ 1. Protect Sensitive Data**

* **Use private for mappings & variables** that should not be exposed.
* Example: mapping (uint256 => Invoice) private invoices; // Prevents exposing invoice details

### **✅ 2. Minimize Function Exposure**

* If a function should only be used inside the contract, **use internal or private** instead of public.

Example: function \_calculateFee(uint256 amount) private pure returns (uint256) {  
 return amount \* 2 / 100; // Internal fee calculation  
}

* **Why private?** No need for external access.

### **✅ 3. Use external for Gas Optimization**

* external functions **use less gas** than public because Solidity does not store extra data for internal calls.
* **Use external if the function is never called inside the contract.**

Example: function processPayment(uint256 invoiceId) external payable {  
 // Payment logic here  
}

* **Why external?** Only external users call this function; no need for internal calls.

### **✅ 4. Use public Wisely**

* Only make a function **public** if you want **both external users & internal functions** to access it.

Example: function getInvoiceDetails(uint256 invoiceId) public view returns (uint256, address, bool) {  
 require(invoiceExists[invoiceId], "Invoice does not exist");  
 Invoice memory inv = invoices[invoiceId];  
 return (inv.amount, inv.payer, inv.isPaid);  
}

* **Why public?** The contract itself & external users may need to check invoice details.

## **🔹 Summary: When to Use Each Modifier**

|  |  |
| --- | --- |
| **Modifier** | **Use Case** |
| public | For getter functions and data that can be visible to everyone |
| private | For internal calculations, mappings, or variables that should not be exposed |
| internal | For functions and variables that should be inherited by child contracts |
| external | For functions that are **only** called from outside the contract (saves gas) |

### **🔹 Best Practices**

✅ **Follow the Principle of Least Privilege** – Only give access to what’s needed.

✅ **Use private & internal wherever possible** to prevent unauthorized access.

✅ **Use external instead of public for gas savings** if a function is never called internally.

✅ **Avoid public state variables** unless necessary, as they generate automatic getter functions.