In Ethereum Virtual Machine (EVM), the difference between storage and memory is fundamental to understanding how data flows in the smart contracts and gas optimization.

	Storage	**Memory**
Data Persistence	Storage is permanent and persists between function calls and transactions. It's where state variables are stored. **Persists indefinitely on the blockchain.**	Memory is temporary and only exists during function execution. It's cleared between external function calls. **Temporary, exists only during function execution.**
Location	Data is stored on the blockchain itself.	Data exists in the executing EVM instance's memory space, not on the blockchain.
Gas Cost	Storage operations are very **expensive** in terms of gas. Writing to storage costs 20,000+ gas, while reading costs 200+ gas.	Memory operations are much cheaper than storage. **Reading/writing costs increase quadratically as memory expands.**
Structure	Storage is organized as a key-value store mapping 256-bit words to 256-bit words.	Memory is a byte array that can be addressed at byte level.
Scope	Storage variables exist at the contract level and are accessible by all functions within the contract.	Memory variables only exist within the function they're declared in.
Declaration	State variables declared at the contract level are automatically stored in storage.	Function parameters and local variables of reference types (arrays, structs) use the memory keyword.

Example	solidity
	`// This variable is stored in storage`
	`uint256 public storedData;`
		solidity
		`function calculate(uint[] memory values) public pure returns (uint[] memory) {`
		`    // Both the parameter and return value are in memory`
		`    uint[] memory results = new uint[](values.length);`
		`    // ...`
		`    return results;`

Data Copying	When assigning storage to storage: creates a reference (points to the same data). When assigning storage to memory: creates a copy	When assigning memory to memory: creates a reference When assigning memory to storage: creates a copy