

# **ЛЕКЦИЯ №4 СМАРТ КОНТРАКТЫ**

Москва, 2020

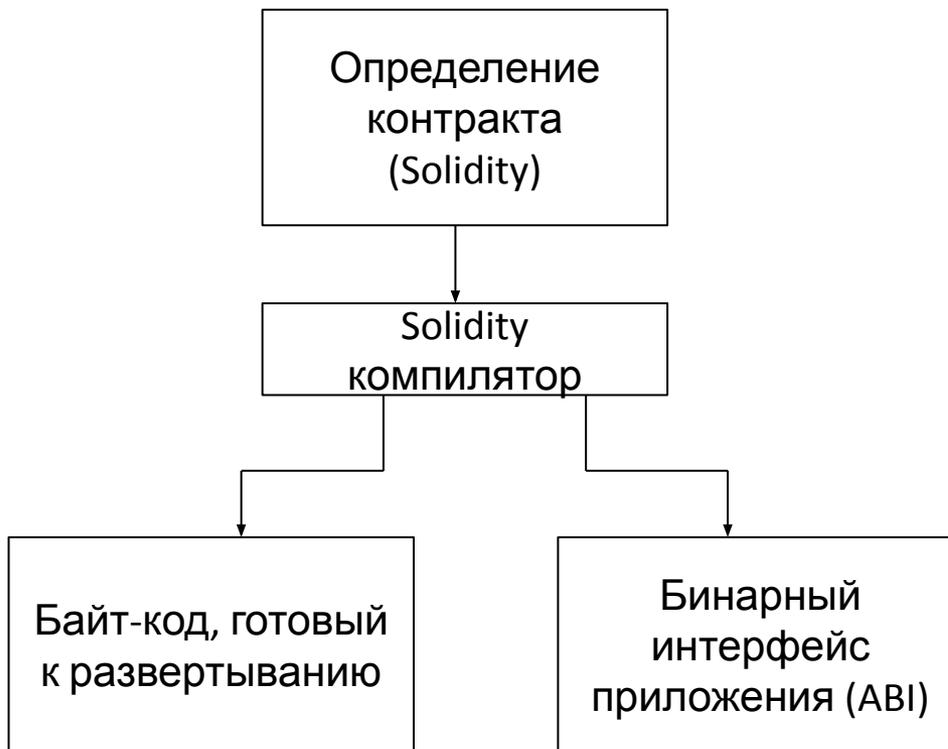
# Язык программирования Solidity

Расширение файлов \*.sol

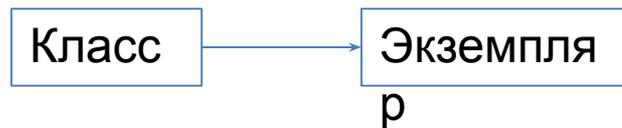
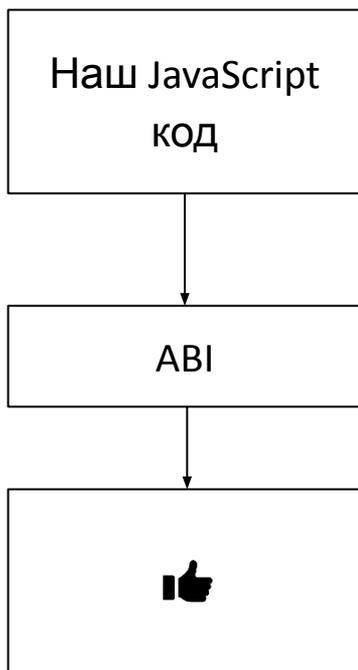
Строгая типизация

Похож на Javascript

## Язык программирования Solidity



## Язык программирования Solidity



## Язык программирования Solidity

Онлайн-редактор кода Remix, созданный специально для создания и тестирования смартконтрактов.

`remix.ethereum.org`

# Язык программирования Solidity

The image shows a screenshot of a Solidity IDE. On the left, a code editor displays the following Solidity code for a contract named 'Inbox':

```
pragma solidity ^0.4.17;

contract Inbox {
    string public message;

    function Inbox(string initialMessage) public {
        message = initialMessage;
    }

    function setMessage(string newMessage) public
    {
        message = newMessage;
    }

    function getMessage() public view returns (string)
    {
        return message;
    }
}
```

On the right, a compilation panel is visible. It has a title bar that says "Switch to the new interface!". The panel shows the current compiler version as "version:0.4.17+commit.bdeb9e52.Emscripten.clang". Below this, there is a dropdown menu labeled "Select new compiler version". There are three checkboxes: "Auto compile" (checked), "Enable Optimization" (unchecked), and "Hide warnings" (unchecked). At the bottom of the panel, there is a button labeled "Start to compile (Ctrl-S)".

At the bottom of the IDE, there is a dropdown menu showing "Inbox" and a "Swarm" button with an upload icon.

## Язык программирования Solidity

The diagram illustrates the components of a Solidity function declaration. Three boxes at the top are labeled "Function name", "Function type", and "Return types". Brackets below these boxes point to the corresponding parts of the code snippet below: "getMessage()" under "Function name", "public view" under "Function type", and "returns (string)" under "Return types".

```
function getMessage() public view returns (string) {  
    return message;  
}
```

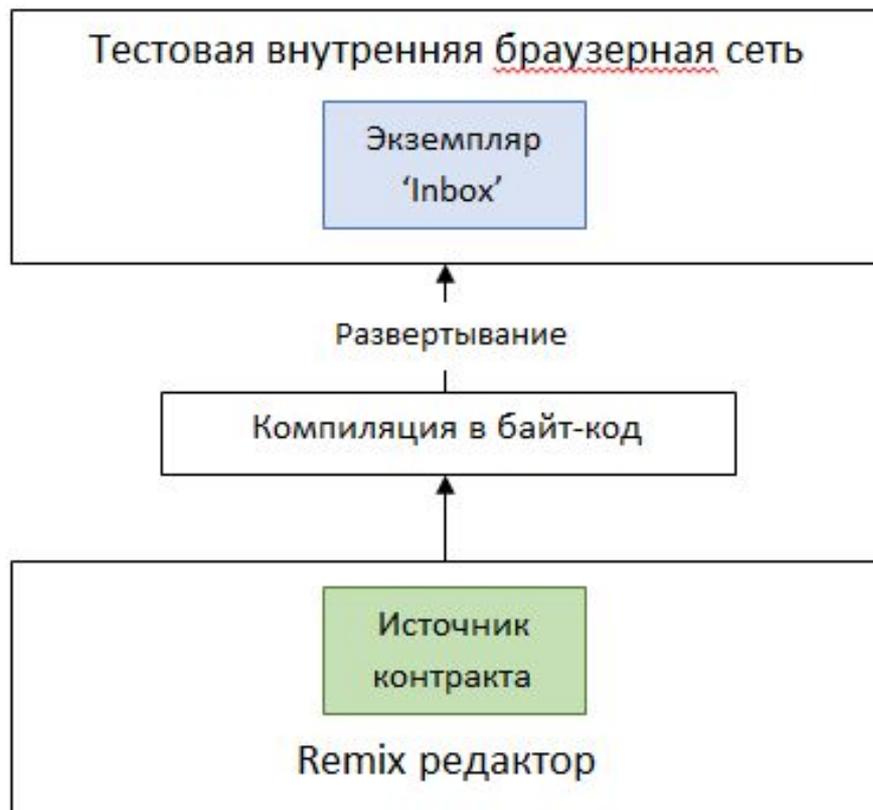
## Язык программирования Solidity

Можно  
использовать  
только один из  
них для функции

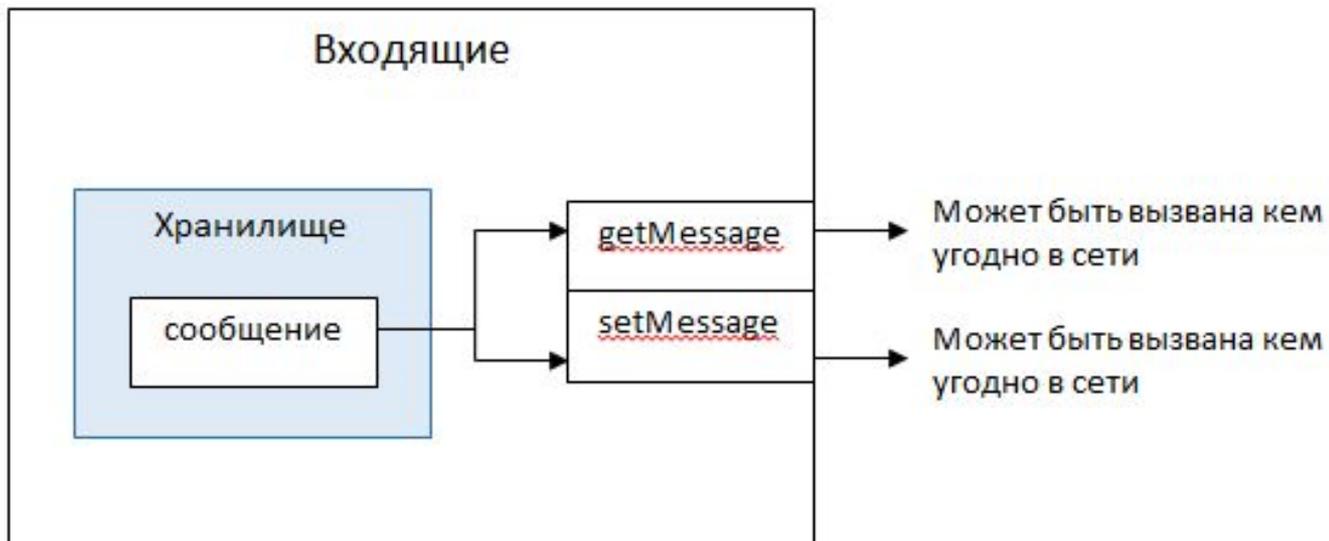
Значат одно и  
то же

Распространённые типы функций	
public	Любой может вызвать эту функцию
private	Только этот контракт может вызывать эту функцию.
view	Эта функция возвращает данные и не изменяет данные контракта
constant	Эта функция возвращает данные и не изменяет данные контракта
pure	Функция не будет изменять или даже считывать данные контракта
payable	Когда кто-то вызывает эту функцию, он может послать эфир

## Язык программирования Solidity



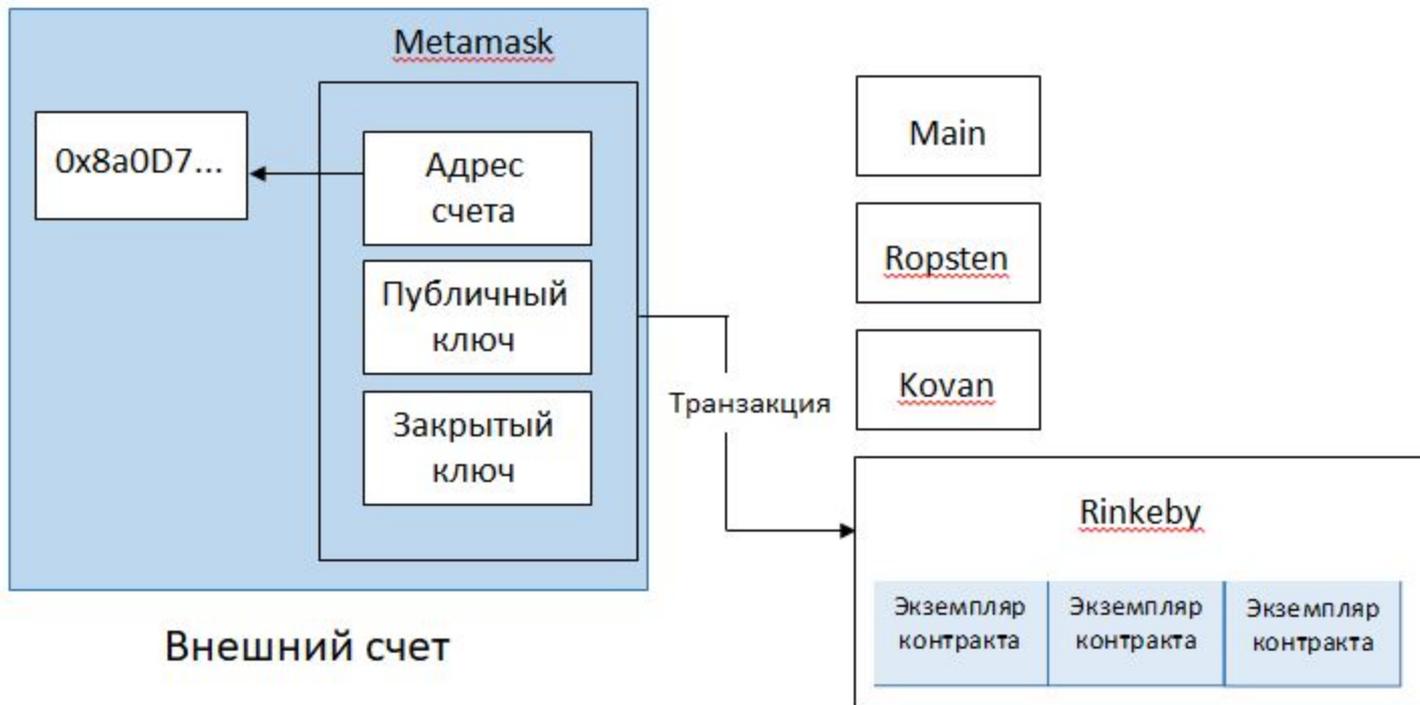
## Язык программирования Solidity



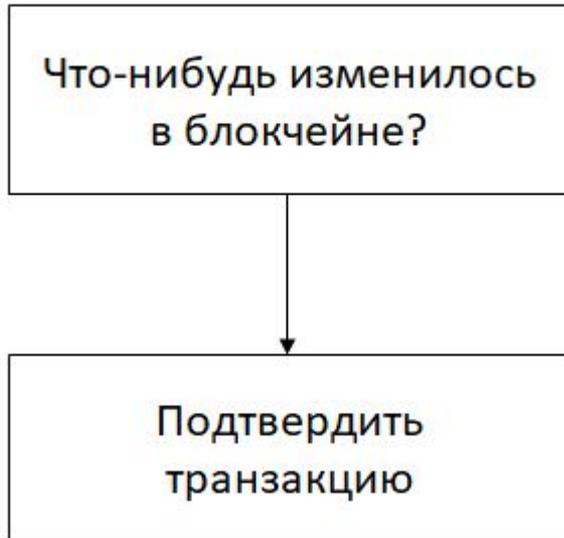
## Язык программирования Solidity

nonce	Сколько раз отправитель проводил транзакцию
to	Адрес счета, на который идут эти деньги
value	Количество эфира, отправляемого на указанный адрес
<u>gasPrice</u>	Количество эфира, которое отправитель готов заплатить за единицу газа, чтобы обработать эту транзакцию
<u>startGas/gasLimit</u>	Предельное количество газа, разрешенное для расходования на выполнение транзакции
v	Криптографические фрагменты данных, которые можно использовать для генерации адреса счета отправителя. Сгенерировано из закрытого ключа отправителя.
r	
s	

# Язык программирования Solidity



## Язык программирования Solidity



Выполнение функций контракта	
Вызов функции	Отправка транзакции в функцию
Невозможно изменить данные контракта	Может изменить данные контракта
Может вернуть данные	Принимает изменения для выполнения!
Работает мгновенно	Возвращает хеш транзакции
Бесплатно!	Стоит денег!

## Язык программирования Solidity

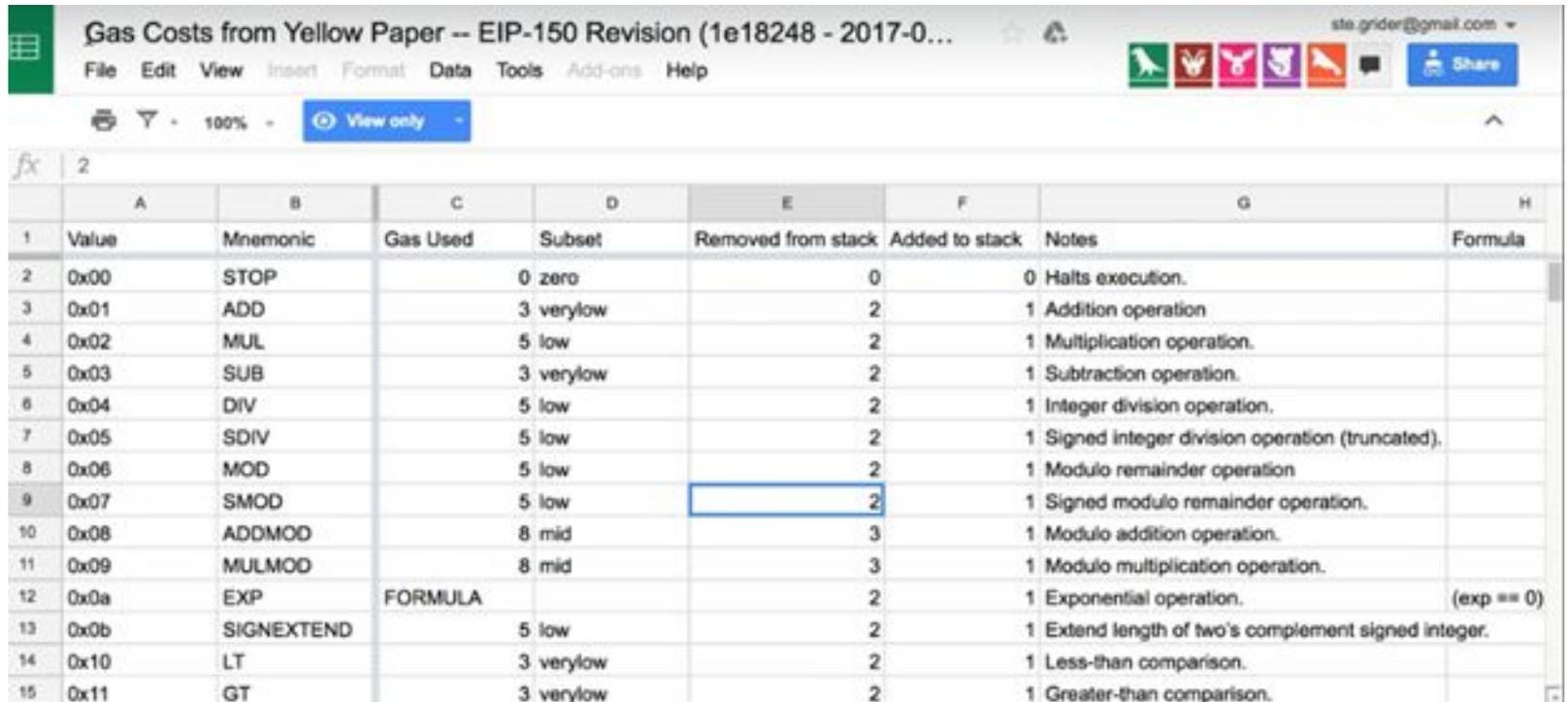


## Язык программирования Solidity

```
13
14 - function doMath(int a, int b) {
15     a + b;
16     b - a;
17     a * b;
18     a == 0;
19 }
20 }
```

<u>gasPrice</u>	Количество эфира, которое отправитель готов заплатить за единицу газа, чтобы обработать эту транзакцию
<u>startGas/gasLimit</u>	Предельное количество газа, разрешенное для расходования на выполнение транзакции

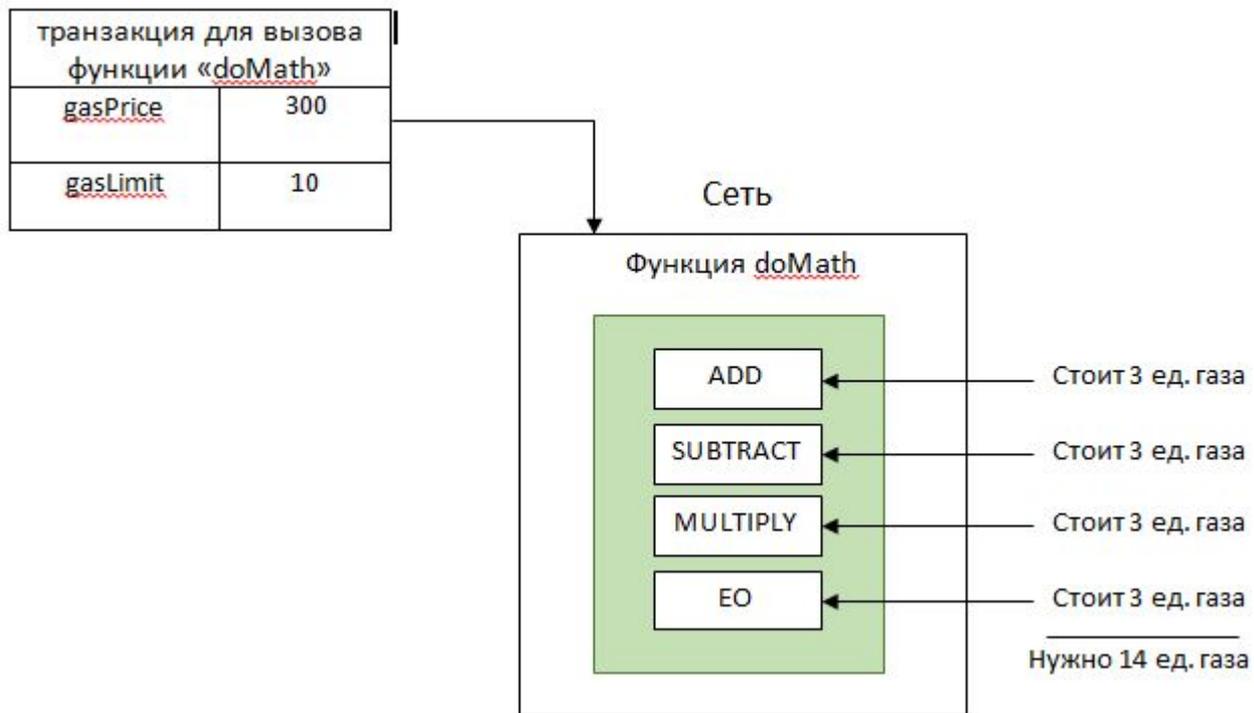
# Язык программирования Solidity



The screenshot shows a Google Sheet with the following data:

	A	B	C	D	E	F	G	H
1	Value	Mnemonic	Gas Used	Subset	Removed from stack	Added to stack	Notes	Formula
2	0x00	STOP		0 zero		0	0 Halts execution.	
3	0x01	ADD		3 verylow		2	1 Addition operation	
4	0x02	MUL		5 low		2	1 Multiplication operation.	
5	0x03	SUB		3 verylow		2	1 Subtraction operation.	
6	0x04	DIV		5 low		2	1 Integer division operation.	
7	0x05	SDIV		5 low		2	1 Signed integer division operation (truncated).	
8	0x06	MOD		5 low		2	1 Modulo remainder operation	
9	0x07	SMOD		5 low		2	1 Signed modulo remainder operation.	
10	0x08	ADDMOD		8 mid		3	1 Modulo addition operation.	
11	0x09	MULMOD		8 mid		3	1 Modulo multiplication operation.	
12	0x0a	EXP	FORMULA			2	1 Exponential operation.	(exp == 0)
13	0x0b	SIGNEXTEND		5 low		2	1 Extend length of two's complement signed integer.	
14	0x10	LT		3 verylow		2	1 Less-than comparison.	
15	0x11	GT		3 verylow		2	1 Greater-than comparison.	

# Язык программирования Solidity



## Язык программирования Solidity

<u>gasPrice</u>	300
-----------------	-----

Использовано 14 ед. газа

$$\text{Итоговая стоимость} = 300 \frac{\text{wei}}{\text{gas}} * 14 \text{ gas} = 4,200 \text{ wei}$$

**!** (logical negation)

**&&**(logical conjunction, “and”)

**||** (logical disjunction, “or”)

**==** (equality)

**!=** (inequality)

---

**BOO**

**L**

Comparisons: `<=`, `<`, `==`, `!=`, `>=`, `>`  
(evaluate to bool)

Arithmetic operators: `+`, `-`, `*`, `/`, `%`, `**`

---

# INTEGER

**<address>.balance**

**(uint256)**: balance of the

Address in Wei

**<address>.transfer(uint256**

**amount)**: send given amount of Wei to  
Address, throws on failure

**<address>.send(uint256 amount)**  
**returns (bool)**:

send given amount of Wei to Address,  
returns false on failure

---

# ADDRES S

Можно хранить UTF-8

Размер динамический

---

**STRING**

**push** - добавить  
элемент

- **length** - узнать /
- переопределить длину
- **new** - не записывать в  
State

---

**ARRAY**

# Mapping - ассоциативный массив

```
mapping(_KeyType =>  
_ValueType)
```

---

# MAPPING

**struct** - способ создать  
собственную структуру данных

Может быть в  
**mapping**

---

# STRUCTURES