**Типы данных**

JS «динамически типизирован» - типы данных есть, но переменные не привязаны ни к одному из них.

**Число.**

Числовой тип данных (number) представляет как целочисленные значения, так и числа с плавающей точкой.

Для типа number безопасным является диапазон чисел ±(253-1). За пределами этого диапазона операции с целыми числами будут небезопасными, и возвращать приближённые значения.

Кроме обычных чисел, существуют так называемые «специальные числовые значения», которые относятся к этому типу данных: Infinity, -Infinity и NaN.

Infinity / -Infinity

Математическая бесконечность ∞. Это особое значение, которое больше любого числа.

Получается в результате деления на ноль, или задается явно.

alert( 1 / 0 ); // Infinity alert( Infinity ); // Infinity

NaN

Означает вычислительную ошибку. Это результат неправильной или неопределённой математической операции. alert( "не число" / 2 ); // NaN

Значение NaN «прилипчиво». Любая математическая операция с NaN возвращает NaN.

Если где-то в математическом выражении есть NaN, то оно распространяется на весь результат. Есть только одно исключение: NaN \*\* 0 равно 1.

**BigInt.**

Позволяет работать с числами произвольной длинны.

Чтобы создать значение типа BigInt, необходимо добавить n в конец числа:

const bigInt = 1234567890123456789012345678901234567890n;

**Строка.**

Между одинарными и двойными кавычками нет разницы. Обратные позволяют встраивать выражения в строку ${…}. Выражение внутри ${…} вычисляется, и его результат становится частью строки.

Нет отдельного типа данных для одного символа.

**Булевый (логический) тип**

…

**Null**

Не относится ни к одному из типов, формирует отдельный тип, который содержит только значение null. Специальное значение, которое представляет собой «ничего», «пусто» или «значение неизвестно».

let age = null; //в приведённом коде указано, что значение переменной age неизвестно.

**Undefined**

Не относится ни к одному из типов, формирует отдельный тип, который содержит только значение undefined. Оно означает, что «значение не было присвоено». Если переменная объявлена, но ей не присвоено никакого значения, то её значением будет undefined.

let age;

alert(age); // выведет "undefined"

Технически можно присвоить значение undefined любой переменной. Но так делать не рекомендуется. Обычно null используется для присвоения переменной «пустого» или «неизвестного» значения, а undefined – для проверок, была ли переменная назначена.

**Symbol**…

…

**Object**

…

**Оператор typeof**

Возвращает тип аргумента строкой.

Есть 2 синтаксические формы:

// Обычный синтаксис

typeof 5 // Выведет "number"

// Синтаксис, напоминающий вызов функции (встречается реже)

typeof(5) // Также выведет "number"

Typeof имеет более высокий приоритет, чем бинарные операторы, по этому если передается выражение, то нужно заключать его в скобки:

typeof 50 + " Квартир"; // Выведет "number Квартир"

typeof (50 + " Квартир"); // Выведет "string"

Результатом вызова typeof null является "object". Это официально признанная ошибка в typeof, ведущая начало с времён создания JavaScript и сохранённая для совместимости.

Вызов typeof alert возвращает "function", потому что alert является функцией. В JS нет специального типа «функция». Функции относятся к объектному типу. Но typeof обрабатывает их особым образом, возвращая "function". Так тоже повелось от создания JavaScript.