**ТИПЫ ДАННЫХ**

В JavaScript типы данных можно разделить на две категории: простые (их также называют примитивными) типы и составные (их также называют ссылочными или объекты).  
К категории простых типов относятся:

* — string — текстовые строки (обычно их называют просто — строки);
* — number — числа;
* — boolean — логические (булевы) значения.

Так же к простым типам относятся два специальных значения:

* — null;
* — undefined

К составным типам данных относятся object — объекты.

String

В JavaScript любые текстовые данные заключенные в двойные или одинарные кавычки являются строками.  
Строки могут состоять из нуля и более символов. Символы включают в себя буквы, цифры, знаки пунктуации, специальные символы и пробелы.

let someText = 'Another text';

let userName = "John Smith";

Строки, заключённые в двойные кавычки, могут содержать символы одиночных кавычек и наоборот:

var text = "It is long 'text'"

var text = 'It is long "text"'

Для того, чтобы в строке, заключённой в двойные кавычки, можно было использовать двойные кавычки, нужно их экранировать с помощью символа \ (обратный слэш), то же самое касается и строки заключённой в одиночные кавычки:

var text = 'It\'s some long'

var text = "It's long \"text\" "

Большие строки можно разбивать на несколько строк, заканчивая каждую строку, кроме последней символом обратного слэша:

"It's some long \

text \

about string"

Длина строки – это количество элементов, содержащихся в ней. Для определения длины строки, используется свойство строки length:

var msg = "hello";

console.log(msg.length); //5

Стандартные, часто используемые методы строк:

str.replace() — Сопоставляет строку с регулярным выражением и заменяет найденную подстроку новой подстрокой.  
str.slice() — Извлекает часть строки и возвращает новую строку.  
str.split() — Разбивает строку на массив подстрок.  
str.toLowerCase() — Преобразует все буквы строки в строчные.  
str.toString() — Преобразует объект в строку.  
str.toUpperCase() — Преобразует все буквы строки в прописные.  
str.includes() — определяет, находится ли одна строка внутри другой, и возвращает true или false, соотвественно.

Number

В Javascript существует только один числовой тип, как для целых так и для дробных чисел:

var count = 15;

var price = 2.5;

Разделителем в дробных числах используется точка.

К числовому типу данных также относятся Infinity/-Infinity(бесконечность) и NaN (Not a Number – не число)

Infinity можно получить путем деления на “0”,

console.log(50 / 0); // Infinity

а NaN является результатом математических операций (кроме операции “+”) со строками.

console.log('is12' \* 2); // NaN

также NaN будет результатом такой операции:

console.log( 0 / 0); // NaN

Значение NaN – единственное , которое не равно ничему, включая себя.

Стандартные, часто используемые методы чисел:

Number.isFinite() — определяет, является ли переданное значение конечным числом, и возвращает true или false, соотвественно.  
Number.isInteger() — определяет, является ли переданное значение целым числом, и возвращает true или false, соотвественно.  
Number.isNaN() — определяет, является ли переданное значение NaN, и возвращает true или false, соотвественно.  
Number.parseFloat() — разбирает строку и возвращает число с плавающей запятой.  
Number.parseInt() — разбирает строку и возвращает целое число.

Boolean

Boolean – логический или булевый тип.

Часто, результат логического выражения внутри условных и других конструкций.  
Также используется для сохранения состояния системы.  
Принимает всего два значения: true (истина) или false (ложь).

var isSelected = true;

var isDataLoaded = false;

Значения true или false указываются без кавычек.

Null

Null – образует свой собственный тип с единственным возможным  
значением null.

var data = null;

Специальное значение, смысл которого «ничего».

Undefined

Также образует свой тип смысл которого: “значение не присвоено”  
Наиболее частый случай: переменная объявлена, но значение ей не присвоено.

var data;

console.log(data); // undefined

В явном виде undefined обычно не присваивают, так как это противоречит его смыслу. Для записи в переменную «пустого» или «неизвестного» значения используется null.

Symbol

Новый (ECMAScript 2015) тип данных, который служит для создания уникальных идентификаторов. В некоторых языках программирования символы также называются атомами.  
Активное использование символов предусмотрено для служебных целей самого javascript.

var newSymbol = Symbol();

var key = Symbol("keyName");

console.log(key); // Symbol(keyName)

У функции, которая создат символ есть один необязательный аргумент, который задает имя символа (часто используется для описания символа, в целях упрощения отладки).

Однако два символа, которые созданы с одним и тем же именем не равны между собой.

console.log(Symbol("key") == Symbol("key")); // false

Symbol предполагается использовать как уникальный ключ для свойств объекта.

var isActive = Symbol("isActive");

var user = {

name : "John",

address : "New York",

[isActive] : true

};

console.log( user[isActive] ); //true

console.log( user.name ); //сработает

console.log( user.isActive ); // ошибка

Для того чтобы не перепутать этот тип с обычным ключем объекта, он записывается с помощью квадратных скобок.  
Обратиться к нему можно также, только с помощью скобок.  
Характерная особенность символа – он не участвует в циклах.

Object

Специальный составной тип, который может содержать данные разного типа.  
Объект может состоять из строк, чисел, логических значений и т.д. Обычно это некая коллекция или набор данных, которые содержат в себе ключ : значение.  
Объявляются объекты при помощи фигурных скобок {…}, например:

var obj = {

id: 15,

name: "John",

active: true

}

Для получения значения объекта, используется следующий синтаксис:

console.log(obj.name);

Также можно вывести все значения объекта вызвав:

console.log(obj) // {id: 15, name: "John", active: true}

typeof

Оператор typeof служит для определения типа.  
Результатом работы typeof является строка, содержащая тип.  
Может быть вызван со скобками или без них.

typeof 1 // "number"

typeof("test") // "string"

typeof(true) // "boolean"

typeof undefined // "undefined"

typeof {} // "object"

Есть одна особенность оператора typeof.

typeof null //object

– это официально признанная ошибка в языке, которая сохраняется для совместимости.