**ГЛОБАЛЬНЫЕ ФУНКЦИИ**

В JavaScript существует множество глобальных функций. Под глобальными функциями понимаются встроенные функции, которые доступны из любой части кода.  
Глобальные функции, которые будут рассмотрены:

* • isNaN
* • isFinite
* • parseInt
* • parseFloat
* • encodeURI
* • encodeURIComponent
* • decodeURI
* • decodeURIComponent

isNaN и isFinite

Функция isNaN() проверяет, является ли переданный параметр числом или нет. Функция вернет true, если параметр не является числом и false, если является.  
Как она работает: переданный параметр преобразуется к числу. Если это не число (строка, массив и т.п.), то он преобразуется в NaN и уже результат этого преобразования будет проверятся на равенство с NaN.  
Стоит учесть, что, к примеру true преобразуется не к NaN, а к числу 1. Подробнее о преобразовании типов данных в соответствующей теме курса.

isNaN('NaN'); //true

isNaN(NaN); //true

isNaN(0 / 0); //true

isNaN(true); // false

isNaN(null); // false

isNaN(10); // false

Кроме глобальной функции isNaN() существует более точная функция Number.isNaN().  
Number.isNaN() не преобразовывает значение в число, по этому в нее надежнее передавать значения, которые обычно превращаются в NaN, но  
на самом деле NaN не являются.

isNaN('NaN'); // true

Number.isNaN('NaN'); // false

isNaN(undefined); // true

Number.isNaN(undefined); // false

Number('NaN'); // NaN

Number(undefined); // NaN

Поскольку isNaN() преобразовывает значение сначала в число, то строка «NaN» и undefined преобразуются в значение NaN.  
В Number.isNaN() преобразование отсутствует, по этому результат более корректный.

isFinite() – определяет, является ли переданное в функцию значение конечным числом (то есть не строкой, массивом и т.п. и не плюс или минус бесконечностью).  
Возвращает false если значение равно NaN или Infinity, в других случаях true.

Number.isFinite(Infinity); // false

Number.isFinite(NaN); // false

Number.isFinite(0); // true

Number.isFinite(444); // true

Как и в случае с isNaN() существует более точная функция Number.isFinite().  
При передаче значения в isFinite(), оно также принудительно будет преобразовано в число и уже после этого проводится сравнение.

Number.isFinite() не преобразовывает значение в число, по этому результат более корректный.

isFinite('0'); // true

Number.isFinite('0'); // false

В случае с isFinite() строковое значение «0» преобразовывается в число, по этой причине мы получаем true.

Number.isFinite() не преобразовывает строковое значение «0» в число, по этому результат будет false, так как строка не может быть конечным числом.

parseInt и parseFloat

Функция parseInt() – выполняет преобразование строки в число посимвольно.  
Эта функция часто используется для получения значений типа ’10px’ — когда вначале стоит число, а потом единицы измерения. Если применить функцию parseInt к ’10px’, то результатом получится число 10 (и это будет действительно число, а не строка).

parseInt('10px', 10); // 10

Вторым параметром можно указать систему счисления числа, и функция вернет число, переведенное из указанной системы счисления в десятичную.

parseInt("15\*3", 10); // 15

parseInt("1.3", 10); // 1

Наткнувшись на символ, который невозможно преобразовать в число, парсинг прекращается и возвращается числовое значение полученное до этого символа.

parseInt('my17px'); // NaN

Преобразование произойдет, если только целое число стоит в начале строки, иначе будет выведено NaN.

Кроме глобально функции parseInt() есть функция Number.parseInt().  
Они абсолютно идентичны.

parseFloat() – преобразовывает строку в десятичное число (число с плавающей точкой).  
Если первый символ не может быть сконвертирован в число, то возвращается NaN.

parseInt('3.14'); // 3

parseFloat('3.14'); // 3.14

encodeURI и encodeURIComponent

URI (Uniform Resource Identifier) — унифицированный (единообразный) идентификатор ресурса.  
Это символьная строка, позволяющая идентифицировать какой-либо ресурс: документ, изображение, файл, службу, ящик электронной почты и т. д.  
URL (Uniform Resource Locator) — единообразный локатор ресурса.  
URL — это URI, который, помимо идентификации ресурса, предоставляет ещё и информацию о местонахождении этого ресурса.

encodeURI() – кодирует универсальный идентификатор ресурса (URI).

Будет закодирована URL строка но не параметры или хеш.

Не кодируются символы:  
; , / ? : @ & = + $ #

Есть символы, которые никогда не кодируются (так называемые не экранируемые символы):  
латинские буквы, десятичные цифры, — \_ . ! ~ \* ‘ ( )

encodeURI("http://www.google.com/results with spaces#some-anchor");

// http://www.google.com/results%20with%20spaces#some-anchor

encodeURIComponent() – кодирует универсальный идентификатор ресурса.  
Будет закодирована URL строка включая специальные символы, которые пропускаются при encodeURI() (кроме не экранированных).

encodeURIComponent("http://www.google.com/results with spaces#some-anchor");

// http%3A%2F%2Fwww.google.com%2Fresults%20with%20spaces%23some-anchor

Таким образом encodeURIComponent() лучше использовать для кодирования параметров URL, а не всего URL полностью.

const param = encodeURIComponent("#some-anchor");

const url = 'http://www.google.com/results/' + param;

console.log(url); // http://www.google.com/results/%23some-anchor

encodeURI() и encodeURIComponent() кодируют специальные символы в URL адресе для их безопасного использования в HTTP запросах.

decodeURI и decodeURIComponent

decodeURI() и decodeURIComponent() используются для соответственного декодирования URL.

decodeURI('http://site.com/%D0%BF%D0%BE%D0%BD%D1%8F%D1%82%D0%BD%D1%8B%D0%B9\_javascript');

// http://site.com/понятный\_javascript

decodeURIComponent('https%3A%2F%2Fdrive.google.com%2Fdrive%2Fmy-drive');

// https://drive.google.com/drive/my-drive