|  |  |
| --- | --- |
| **ПРЕОБРАЗОВАНИЕ ТИПОВ МОЖЕТ БЫТЬ ЯВНЫМ ИЛИ НЕЯВНЫМ**   * value = Number('23') // явное преобразование * value = 5 + '25' // неявное преобразование   ПРОВЕРКА ТИПА ДАННЫХ  console.log(typeof value); | |
| **ПРЕОБРАЗОВАНИЕ СТРОКИ**  String(123) // '123'  String(-12.3) // '-12.3'  String(null) // 'null'  String(undefined) // 'undefined'  String(true) // 'true'  String(false) // 'false'  String(function () {}) // 'function () {}'  String({}) // '[object Object]'  String({ key: 42 }) // '[object Object]'  String([]) // ''  String([1, 2]) // '1,2' | **ПРЕОБРАЗОВАНИЕ БУЛЕВЫХ ЗНАЧЕНИЙ**  Boolean('') // false  Boolean('string') // true  Boolean('false') // true  Boolean(0) // false  Boolean(42) // true  Boolean(-42) // true  Boolean(NaN) // false  Boolean(null) // false  Boolean(undefined) // false  Boolean(function () {}) // true  Boolean({}) // true  Boolean({ key: 42 }) // true  Boolean([]) // true  Boolean([1, 2]) // true |
| **ПРЕОБРАЗОВАНИЕ ЧИСЕЛ**  Number('123') // 123  Number('123.4') // 123.4  Number('123,4') // NaN  Number('') // 0  Number(null) // 0  Number(undefined) // NaN  Number(true) // 1  Number(false) // 0  Number(function () {}) // NaN  Number({}) // NaN  Number([]) // 0  Number([1]) // 1  Number([1, 2]) // NaN | **parseInt и parseFloat**  Функция parseInt возвращает целое число, а parseFloat возвращает число с плавающей точкой:  alert( parseInt('100px') ); // 100  alert( parseFloat('12.5em') ); // 12.5  alert( parseInt('12.3') ); // 12, вернётся только целая часть  alert( parseFloat('12.3.4') ); // 12.3, произойдёт остановка чтения на второй точке  alert( parseInt('a123') ); // NaN, на первом символе происходит остановка чтения  alert( parseInt('0xff', 16) ); // 255  alert( parseInt('ff', 16) ); // 255, без 0x тоже работает  alert( parseInt('2n9c', 36) ); // 123456 |
| **Неявное преобразование типов**  Неявное преобразование происходит, когда мы заставляем JavaScript работать со значениями разных типов. Например, если мы хотим «сложить» число и строку  Примитивный тип данных  Интерпретатор приведёт примитивные значения к логическим, если мы используем && или ||.  К строке, если мы используем +, когда один из операндов — строка.  К числу, если:  мы используем операторы сравнения <, <=, >, >=;  используем арифметические операции -, + (за исключением пункта 2), /, \*.  используем унарный плюс: +'2' === 2;  используем оператор нестрогого сравнения ==.  Интерпретатор приводит их к логическому, если мы используем && или ||. Объекты — всегда true.  С числом и строкой всё немного интереснее. Чтобы определить, к строке приводить значение или к числу, JavaScript смотрит, какой из двух методов (valueOf() и toString()) в текущем объекте объявлен.  Если перед нами не объект Date, то метод valueOf() вызывается, обычно, первым (если не сильно углубляться в детали спецификации).  Если возвращённое после этого значение — это примитив, то возвращается оно.  Если нет, то вызывается другой метод (если valueOf() не вернул примитив, то вызывается toString() и наоборот).  Если после этого вернулся примитив, возвращается он.  Если даже после этого не вернулся примитив, то будет ошибка Uncaught TypeError: Cannot convert object to primitive value. | |