|  |  |  |  |
| --- | --- | --- | --- |
|  | **Код** | **Выполнение** | **Объяснение** |
| 1 | **var str = "Учим Ява Скрипт"**  **str.search("Ява")** | 5 | search() ищет совпадения между выражением и строкой текста и возвращает позицию найденного совпадения (если оно имеется).  Данный метод возвращает -1 если совпадение не было найдено. |
| 2 | **var arr = str.split([separator][, limit])** |  | **separator**  регулярное выражение или строка, по которой делить str  **limit**  максимальное количество кусков, на которые может быть разбита строка  Метод split возвращает новый массив. Строка бьется по separator, при разбивании separator пропадает:  arr = "a,b,c".split(',')  // массив ["a", "b", "c"]  Если separator не указан или совпадений нет, то возвращаемый массив состоит из одного элемента - исходной строки. |
| 3 | **String.fromCharCode(num1, ..., numN)** |  | **num1, ..., numN**  последовательность чисел, представляющих собой коды Unicode-символов  String.fromCharCode(65,66,67) // вернет строку "ABC".  Создание строки по таблице  символов Unicode |
| 4 | **var codepoint = string.charCodeAt(index)** |  | **index**  целое число от 0 до длины строки-1  Юникодное значение от 0 до 1,114,111. Первые 128 значений Unicode совпадают с кодировкой ASCII.  Заметим, что charCodeAt() всегда возвращает значение, меньшее 65536.   |  | | --- | | // возвратит 65, unicode-код для А |  |  | | --- | | "ABC".charCodeAt(0) // 65 | |
| 5 | **result = string1.concat(string2, string3[, ..., stringN])** |  | **string2...stringN**  строки, которые будут прибавлены к string1  Выполняет действие, аналогичное оператору + для строк. |
| 6 | **str.valueOf()** |  | Метод valueOf возвращает элементарное значение для объекта [String](http://javascript.ru/string). Это значение эквивалентно [String#toString](http://javascript.ru/String/toString).   |  | | --- | | x= new String("Hello world"); | | alert(x.valueOf()) // покажет"Hello world" | | |
| 7 | **str.substring(indexA, [indexB])** |  | **indexA**  целое число от 0 до длины строки-1  **indexB**  целое число от 0 до длины строки-1  Метод substring возвращает подстроку, начиная с позиции indexA до, но не включая indexB.  В частности:   * Если indexA = indexB, возвращается пустая строка * Если indexB не указан, substring возвращает символы до конца строки * Если какой-то из аргументов меньше 0 или является NaN - он считается равным 0 * Если какой-то из аргументов больше, чем длина строки - он считается равным длине строки   Если indexA > indexB, тогда substring ведет себя, как будто аргументы поменялись местами.  Например, str.substring(1, 0) == str.substring(0, 1) |
| 8 | **var sub = string.substr(start[, length]** |  | **start**  начальная позиция, начиная от 0  **length**  сколько символов брать с позиции start  Этот метод описан только в ненормативной части стандарта ECMA, но реализован во всех браузерах.  Метод substr начинает собирать строку с позиции start и заканчивает, когда соберет length или дойдет до конца строки.  Если start больше или равен длине строки, то substr возвращает пустую строку.  Если start < 0, позиция определяется с конца строки. При этом если полученная позиция - до начала строки, то в качестве start берется 0. Это поведение не поддерживается Internet Explorer.  Если параметр length не указан - подстрока берется до конца строки.  Если length <= 0 - возвращается пустая строка. |
| 9 | **str.replace(regexp, newSubStr|function)** |  | **regexp**  объект типа [RegExp](http://javascript.ru/RegExp) или строка  **newSubStr|function**  новая подстрока или функция, которая ее возвращает  Этот метод не меняет вызывающую строку, а возвращает новую, после замен.   |  | | --- | | str = "тест еще тест" |  |  | | --- | | str.replace("тест","прошел") // = "прошел еще тест" | |
|  |  |  |  |