JavaScript

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sl no | Function Name/Operator | Description | Example |  |
| 1 | Exponentiation operator | The exponentiation operator (\*\*) will return the first operand’s power of the second operand. | a \*\* b is equivalent to ab, which is equivalent to Math.pow(a, b) | console.log(10 \*\* 2); // 100 |
| 2 | Length | The length property returns the length of a string: | let text = "ABCDEFGHIJ";  let length = text.length; |  |
| 3 | at(),charAt(),[ ] | This method returns the character at a specified index (position) in a string | let text = "HELLO WORLD";  let char = text.charAt(0); |  |
| 4 | charCodeAt() | The charCodeAt() method returns the code of the character at a specified index in a string | let text = "HELLO WORLD";  let char = text.charCodeAt(0); |  |
| 5 | Slice() | slice() extracts a part of a string and returns the extracted part in a new string | let text = "Apple, Banana, Kiwi";  let part = text.slice(7, 13); | Slice out a portion of a string from position 7 to position 13: |
| 6 | substring() | substring() is similar to slice() | let str = "Apple, Banana, Kiwi";  let part = str.substring(7, 13); | The difference is that start and end values less than 0 are treated as 0 in substring(). |
| 7 | substr() | The difference is that the second parameter specifies the length of the extracted part. | let str = "Apple, Banana, Kiwi";  let part = str.substr(7, 6); |  |
| 8 | toLowerCase() | A string is converted to upper case | let text1 = "Hello”  let t2 = t1.toLowerCase(); |  |
| 9 | toUpperCase() | A string is converted to upper case | let t1 = "Hello";  let t2 = t1.toUpperCase(); |  |
| 10 | trim() | The trim() method removes whitespace from both sides of a string | let t1 = " Hello ";  let t2 = t1.trim(); |  |
| 11 | trimStart() | The trimStart() method works like trim(), but removes whitespace only from the start of a string. | let t1 = " Hello";  let t2 = t1.trimStart(); |  |
| 12 | trimEnd() | The trimEnd() method works like trim(), but removes whitespace only from the end of a string. | let t1 = "World! ";  let t2 = t1.trimEnd(); |  |
| 13 | padStart() | The padStart() method pads a string from the start. | let text = "5";  let padded = text.padStart(4,"0"); | Pad a string with "0" until it reaches the length 4: |
| 14 | padEnd() | The padEnd() method pads a string from the end. | let text = "5";  let padded = text.padEnd(4,"0"); | The padEnd() method is a string method.  To pad a number, convert the number to a string first.  let numb = 5;  let text = numb.toString();  let padded = text.padEnd(4,"0"); |
| 15 | concat() | concat() joins two or more strings | let t1 = "Hello";  let t2 = "World";  let t3 = t1.concat(" ", t2); |  |
| 16 | repeat() | The repeat() method returns a string with a number of copies of a string.  The repeat() method returns a new string.  The repeat() method does not change the original string. | let t1 = "Hello world!";  let result =t1.repeat(4); | Hello world!Hello world!Hello world!Hello world! |
| 17 | replace() | The replace() method returns a new string.  The replace() method replaces only the first match.  By default, the replace() method is case sensitive | let t1 = "This is table”  let nT = t1.replace("This", "It"); | If you want to replace all matches, use a regular expression with the /g flag set.  To replace case insensitive, use a regular expression with an /i flag |
| 18 | ReplaceAll() | In 2021, JavaScript introduced the string method replaceAll() | let t= "I am Mamun. I go to school."  t= text.replaceAll("I","He"); |  |
| 19 | indexOf() | The indexOf() method returns the index (position) of the first occurrence of a string in a string | Let t1 = “Mamun is a good boy”  Let t2 = t1.indexOf(‘is’)  o/p = 6 | it returns -1 if the string is not found:  JavaScript counts positions from zero.  Accept a second parameter as the starting position for the search |
| 20 | includes() | The includes() method returns true if a string contains a specified value.  Otherwise it returns false | Let t1 = “I go to mosque”  Let t2 = includes(“to”8); | includes() is case sensitive.  This example return false, and Check Start at position 12: |
| 21 | startsWith() | The startsWith() method returns true if a string begins with a specified value | Let t1 = “I am here”  Let t2 = startsWith(“He”) | Return false |
| 22 | endsWith() | Otherwise it returns false:  The endsWith() method returns true if a string ends with a specified value. | Let t1 = “I am here”  Let t2 = **endsWith**(“He”) | Return false |
| 23 | toString() | you can use the toString() method to output numbers from base 2 to base 36.  Hexadecimal is base 16. Decimal is base 10. Octal is base 8. Binary is base 2. | let myNumber = 32;  myNumber.toString(32);  myNumber.toString(10);  myNumber.toString(8);  myNumber.toString(2); | toString()-The toString() method returns a number as a string. |
| 24 | toPrecision() | toPrecision() returns a string, with a number written with a specified length | Let t1=100.55 | T1.toPrecision(3);  Return 100 |
| 25 | Number.isInteger() | The Number.isInteger() method returns true if the argument is an integer. | Number.isInteger(10) |  |
| 26 | Number() | Returns a number converted from its argument. | Number("10") | Return 10 |
| 27 | Number.isSafeInteger() | Returns true if the argument is a safe integer | Number.isSafeInteger(12345678901234567890) | Return false |
| 28 | Number.parseFloat() | Converts a string to a number | Number.parseFloat("10") | Return 10 |
| 29 | Number.parseInt() | Converts a string to a whole number | Number.parseInt("-10") | Return -10 |
| 30 | splice() | The splice() method adds new items to an array | const fruits = ["Banana", "Orange", "Apple", "Mango"];  fruits.splice(2, 0, "Lemon", "Kiwi"); | The first parameter (2) defines the position **where** new elements should be **added**  The second parameter (0) defines **how many** elements should be **removed**.  The rest of the parameters ("Lemon" , "Kiwi") define the new elements to be **added**. |
| 31 | toSpliced() | toSpliced() method creates a new array, keeping the original array unchanged | let name = ['Mamun', 'Hossen', 'Melon', 'Khan'];  let afterSplice = name.toSpliced(2, 2, "ena", "ben");  console.log(afterSplice);  console.log(name); | After toSplice ['Mamun', 'Hossen', 'ena', 'ben']  **Orginal value**  ['Mamun', 'Hossen', 'Melon', 'Khan'] |