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'] |
| 32 | slice() | Slice method একটি array থেকে slice করে নতুন একটি array তৈরী করে। slice method দুটি parameter নেয়। first parameter mendatory. Second Parameter omitted. | let name = ['Mamun', 'Hossen', 'Melon', 'Khan'];  let after\_slice = name.slice(1,3);  first parameter হচ্ছে কোন পজিশন থেকে নতুন array শুরু হবে। second parameter হচ্ছে কোন পজিশন এ গিয়ে শেষ হবে। |  |
| 33 | join() | Join method single array এর ক্ষেত্রে element এর মধ্যে একটি string/number জয়েন করে দেয় এবং multidimensional array এর ক্ষেত্রে প্রত্যেক array এর মধ্যে string/number জয়েন করে দেয়। | let number = [9, 9];  let after\_join = number.join(' # ');  console.log(after\_join); | 9 # 9 |
| 34 | pop() | pop() array থেকে শেষ এর একটি elemdent delete করে। | let fruits =["a", "b", "c", "d"];  fruits.pop();  console.log(fruits); | (3) ['a', 'b', 'c'] |
| 35 | push() | push() একটি array এর শেষে নতুন একটি elemdent add করে। | let fruits = ["a", "b", "c", "d"];  fruits.push('e');  console.log(fruits); | (5) ['a', 'b', 'c', 'd', 'e'] |
| 36 | sort() | sort() একটি single array কে alphabetically শর্ট করে। | let fruits = ['m','a','d','b'];  fruits.sort(); | manin value 10,3,6,7  after sort 10,3,6,7 |
| 37 | reverse() | reverse() একটি single array এর element কে reverse করে। | let number = [6, 7, 3, 10];  number.reverse();  console.log(number); | log [10, 3, 7, 6] |
| 38 | toReversed() | toReversed() একটি single array এর element কে reverse করে। তবে original array পরিবর্তন করেনা | let number = [6, 7, 3, 10];  let reverse\_value = number.toReversed();  console.log(`main value ${number}`);  console.log(`after reverse ${reverse\_value}`); | main value 6,7,3,10  after reverse 10,3,7,6 |
| 39 | toSorted() | toSorted () একটি single array কে alphabetically শর্ট করে। তবে original array পরিবর্তন করেনা | let number = [6, 7, 3, 10];  let sort\_value = number.toSorted();  console.log(`main value ${number}`);  console.log(`after sort ${sort\_value}`); | Log(main value) -6,7,3,10  Log (after sort)-10,3,6,7 |
| 40 |  |  | let n = [12, 4, 25, 3, 45];  let after\_sort = n.sort(function(a, b) {  return a - b  });  console.log(`after sort ${after\_sort}`); | after sort 3,4,12,25,45 |
| 41 | Math.min.apply | Math.min.apply একটি array থেকে lowest value কে find করে। | let n = [12, 4, 25, 3, 45];  console.log(Math.min.apply(null, n)); | Log 3  Math.min.apply(null, [1, 2, 3]) is equivalent to Math.min(1, 2, 3). |
| 42 | Math.max.apply | Math.max.apply একটি array থেকে highest value কে find করে। | let n = [12, 4, 25, 3, 45];  console.log(Math.max.apply(null, n)); | Log 45 |
|  |  |  | Array Iteration Methods |  |
| 43 | forEach() | forEach()একটি function কে কল করে(callback function) প্রত্যেকটি array element এর জন্য। | let n = [12, 4, 25, 3, 45];  n.forEach(test);  function test(value){  console.log(value);  } | 12 4 25 3 45 |
| 44 | map() | map() নতুন একটি array create করে প্রত্যেকটি array element এর জন্য। | let n = [12, 4, 25, 3, 45];  let custom\_arr = n.map(function(x) {  return x\*2;  });  console.log(n);  console.log(custom\_arr); | (5)[12, 4, 25, 3, 45]  (5) [24, 8, 50, 6, 90] |
| 45 | flatMap() | flatMap() নতুন একটি array create করে প্রত্যেকটি সকল array element এর জন্য। | let n = [  [9, 4],  [3, 4]  ];  let value = n.flatMap((x) => x.flatMap((y) => y > 4));  console.log(value); | Log [true, false, false, false] |
| 46 | reduce() | reduce() একটি array এর প্রতিটি element এর জন্য একটি function রা্ন করে এবং একটি single value return করে।  reduce() একটি array এর left থেকে right এর দিকে কাজ করে। | let number = ['4', '5', '6'];  let result = number.reduce((total, value) => Number(total) + Number(value));  console.log(result); | result 15  Total=initial value  Value = item value  Note that the function takes 4 arguments: |
| 47 | reduceRight() | reduceRight()একটি array এর প্রতিটি element এর জন্য একটি function রা্ন করে এবং একটি single value return করে।  reduceRight() একটি array এর right থেকে left এর দিকে কাজ করে। | let number = ['4', '5', '6'];  let result = number.reduceRight((total, value) => Number(total) + Number(value));  console.log(result); | result 15 |
| 48 | every() | every() একটি ‍array এর প্রত্যেকটি element কে condition wise check করে এবং সকল element, condition true করলে ture অন্যথায় false return করে। | let number = [[9, 4], [5, 7] ];  let number\_two = number.flat();  let result = number\_two.every(function(x) {  return x > 2;  }) | Return true |
| 49 | some() | some() একটি ‍array এর element কে condition wise check করে এবং কোন একটি element, condition true করলে ture অন্যথায় false return করে। | let number = [4, 9];  let result = number.some(function(x) {  return x > 4;  })  console.log(result); | Return ture |
| 50 | Array.from() | Array.from() | let number = 'abh';  let result = Array.from(number);  console.log(result); | Log  ['a', 'b', 'h'] |
| 51 | with() | with() মূল অ্যারে পরিবর্তন না করে অ্যারের উপাদানগুলি আপডেট করার নিরাপদ উপায় । | let number = ['2', ' 4', '5', ' 6'];  let result = number.with('2','Five');  console.log(result); | Log ['2', ' 4', 'Five', ' 6'] |
| 52 | Math.round() | Math.round() কাছের integer রিটার্ন করে। | let num = 4.2;  console.log(`after round ${Math.round(num)}`); | Log after round 4 |
| 53 | Math.ceil() | Math.ceil() উপরের interger return করে। | let num = 4.2;  console.log(`after ceil ${Math.ceil(num)}`); | Log after round 5 |
| 54 | Math.floor() | Math.floor() নিচের interger return করে। | let num = 4.2;  console.log(`after floor ${Math.floor(num)}`); | Log after floor 4 |
| 55 | Math.pow(x, y) | Math.pow(x, y) এখানে y হচ্ছে x এর power | let num = 4;  let num1 = 2;  console.log(`after pow ${Math.pow(num,num1)}`); | after pow 16 |
| 56 | Math.sqrt() | Math.sqrt(x) স্কয়ার রুট রিটার্ন করে। | let num = 4;  console.log(`after square root ${Math.sqrt(num)}`); | after square root 2 |
| 57 | Math.abs(x) | Math.abs(x) পজেটিভ value return করে। | let num = -4;  console.log(`after absolute value ${Math.abs(num)}`); | after absolute value 4 |
| 58 | Math.min() and Math.max() | Math.min() / Math.max() একটি লিস্টের মধ্য থেকে ছোট /বড় ডাটা বের করে। | let num = [2, 5, 7, 4, 1, 9];  console.log(`result ${Math.min(4, 1, 9)}`); | Log result 1 |
| 59 | Math.random() | Math.random() একটি random value return করে ০ এবং 1 এর মধ্যে । | console.log(` random value ${Math.random()}`); | Log random value 0.5139041507564441 |
| 60 | for of | Arrays, Strings, Maps, NodeLists কে iterate করার জন্য। | const person = [  ['one', 'two', 'three'],  ['four', 'Five', 'Six'],  ['Seven', 'Eight', 'Nine']  ];  let txt = "";  for (let x of person) {  for (let y of x) {  console.log(y);  }  } | one  two  three  four  Five  Six  Seven  Eight  Nine |
| 61 | for in | Object কে iterate করার জন্য | const person = [{  Name: 'Mamun',  age: 12,  roll: 10  },  {  Name: 'Hanna',  age: 5,  roll: 3  }  ]  for (let x of person) {  for (let y in x) {  console.log(x[y]);  }  } | Log  Mamun  12  10  Hanna  5  3 |
| 62 | Strict mode |  | Variable declare না করে কোন veriable use করা যাবেনা। |  |
| 63 | Rest Parameter | Function এর শেষের  Parameter এর পিছনে তিনটি ডট দিলে সেটাকে rest parameter বলা হয়। function call করার সময় মূল parameter এর value বাদে যতগুলো দেয়া হয় সবগুলো rest parameter এ ধারন করে। | function test(a, b, ...c) {  console.log(`result ${a} and ${b} and ${c}`);  }  test(2, 3, 4, 5, 6, 7); | Log result 2 and 3 and 4,5,6,7 |
| 64 | XMLHttpRequest() | Event-onload(),onerror() | property- Response,responseText,responseType,responseurl,status,statustext | Function – open(),send(),setRequestHeader() |
| 66 | Fetch() | Fetch() method দ্বারা XMLHttpRequest() কে replace করা হয়েছে। Fetch() হচ্ছে global method. | Fetch() কে দুই ভাবে কল করা যায়। then, asyn await. Fetch() by default promise return করে। Promise can only handle network error |  |
| 67 | PUT,PATCH |  | UPDATE করার জন্য PUT ব্যবহার করা হয়।  Single একটি element update করার জন্য PATCH ব্যবহার করা হয়। |  |