Metaverse

AI

VR / AR

Blockchain

1. Slice and splice

2. forEach and Map

3. in operator

4. Getter and Setter

5. Iterate on Java Object

Resize array using array.length

|  |
| --- |
| a=['Pune','Hyderabad','Banglore','Mumbai','Indore','Delhi'] |
|  | console.log(a.length) //OUTPUT 6 |
|  | a.length=3 |
|  | console.log(a) //OUTPUT ['Pune','Hyderabad','Banglore'] |

Swapping 2 numbers

|  |
| --- |
| let a=10; |
|  | let b=20; |
|  | console.log(a,b) //OUTPUT: 10,20 |
|  | [a,b]=[b,a] |
|  | console.log(a,b) //OUTPUT :20,10 |

Concatenating 2 or more arrays without causing server overload

|  |
| --- |
| //new way |
|  | a=[1,2,3,4,5] |
|  | b=[4,5,6,7,8] |
|  | a.push.apply(a,b) // it will only push the content of array b in array a. |
|  | console.log(a) |

4. Use Filter in different way

|  |
| --- |
| a=[null,undefined,{"name":"Alex"},{"name":"Nick"},{"name":""},null] |
|  | c=a.filter(item=>item.name.length>0) //this will show error Uncaught TypeError: Cannot read property 'length' of undefined |
|  |  |
|  | // we can use filter with Boolean to remove null and undefined Values |
|  |  |
|  | c=a.filter(Boolean).filter(it=>it.name.length>0) //This will work |
|  | console.log(c) //OUTPUT |

5. Iterate on map from 0 to n

[...Array(100)].map((it,index)=>console.log(index))

6. Replace all occurrence of word in String

|  |
| --- |
| str="India India USA India UK India" |
|  | console.log(str.replace(/India/,'USA')) |
|  | //OUPUT USA India USA India UK India |
|  | //Add g at the end of string it will replace all occurences |
|  | console.log(str.replace(/India/g,'USA')) |

7 Shortcut for condition

|  |
| --- |
| // old way |
|  | let a="hello" |
|  | let b; |
|  | if(a==undefined) |
|  | { |
|  | b="Nothing" |
|  | } |
|  | else |
|  | { |
|  | b=a |
|  | } |
|  | console.log(b) //hello |
|  |  |
|  | //new way |
|  | b=a||"Nothing" |
|  | console.log(b) //hello |
|  |  |
|  | // old way |
|  | let data={"name":"Allen"} |
|  | if(data!=undefined)console.log(data.name) |
|  |  |
|  | // new way |
|  | data!=undefined&&console.log(data.name) |

8 String to number / Number to String

|  |
| --- |
| // string to number |
|  | a="123" |
|  | console.log(+a) //Output 123 |
|  | b="" |
|  | console.log(+b) //NaN |
|  |  |
|  | //number to string |
|  |  |
|  | a=123 |
|  | console.log(a+"") |

9 Use the console in different ways

|  |
| --- |
| // %s replaces an element with a string |
|  | console.log("Helllo I love %s","Javascript") |
|  | // %d replaces an element with an integer |
|  | console.log("Helllo %d ",1) |
|  | // %f replaces an element with a float |
|  | console.log("Helllo %f ",1.078) |
|  | // %(o|O) | element is displayed as an object. |
|  | console.log("Helllo %O",{"Name":"Sidd"}) |
|  | // %c | Applies the provided CSS |
|  | console.log("%cThis is a red text", "color:red"); |

10. Use console.table

|  |
| --- |
| // we can use console.table to show objects in tabular format |
|  | a=[{"city":"Banglore"},{"city":"Pune"},{"city":"Mumbai"}] |
|  | console.table(a) |

11. Get item near to last of an array

|  |
| --- |
| a=[1,2,3,4,5,6,7,8,9,10] |
|  | conosle.log(a.slice(-1))//[10] |
|  | console.log(a.slice(-2))//[9,10] |

12. Get n power of number

|  |  |
| --- | --- |
|  | console.log(2 \*\* 3) //8 |
|  | console.log(2 \*\* 12) //4096 |

13. Check the falsy and truly vlue

|  |  |
| --- | --- |
|  | //In javascript following values are falsy 0, "", null, undefined, NaN and of course false except it all are truly |
|  | // use !! operator to get falsy or trult |
|  | console.log(!!0,!! "",!!null,!! "Hii",!! undefined,!! NaN ,!! false,!! true) |
|  | //OUTPUT false false false true false false false true |

14. Call a function by its name stored in string using eval function

|  |
| --- |
| const print=()=>console.log("hello") |
|  | setTimeout(()=>eval('print')(),1000)//hello |

15. typeof Operator

|  |
| --- |
| //The typeof operator returns a string indicating the type of the unevaluated operand. |
|  | console.log(typeof 12) //number |
|  | console.log(typeof "hello") //string |
|  | console.log(typeof []) //object |
|  | console.log(typeof true) //boolean |
|  | console.log(typeof {}) //object |

16 yield keyword

|  |
| --- |
| //The yield keyword is used to pause and resume a generator function |
|  | function\* list(index,length) { |
|  | while (index < length) { |
|  | yield index; |
|  | index++; |
|  | } |
|  | } |
|  |  |
|  | const iterator = list(0,10); |
|  |  |
|  | console.log(iterator.next().value); |
|  | // expected output: 0 |
|  |  |
|  | console.log(iterator.next().value); |
|  | // expected output: 1 |

17 function in javascript

|  |  |
| --- | --- |
|  | //The function\* declaration (function keyword followed by an asterisk) defines a generator function, which returns a Generator object. |
|  | function\* generator(i) { |
|  | yield i; |
|  | yield i + 10; |
|  | } |
|  |  |
|  | const gen = generator(10); |
|  |  |
|  | console.log(gen.next().value); |
|  | // expected output: 10 |
|  |  |
|  | console.log(gen.next().value); |
|  | // expected output: 20 |

18. new.target in javascript

|  |  |
| --- | --- |
|  | // new.target is used to detect that weather a function or constructor call using new or not. |
|  | // if it called using new it will return refrence to constructor or function else it return undefined. |
|  | function learn(){ |
|  | new.target?console.log("Called using new"):console.log("Called without new") |
|  | } |
|  |  |
|  | learn() //called without learn |
|  | new learn() //called using new. |
|  | //In arrow functions, new.target is inherited from the surrounding scope. |

19. label statement

|  |
| --- |
| //The labeled statement can be used with break or continue statements. It is prefixing a statement with an identifier which you can refer to. |
|  | loop1: |
|  | for (let i = 0; i < 5; i++) { |
|  | loop2: |
|  | for(let j=0;j<5;j++) |
|  | { |
|  | console.log(i,j) |
|  | if(i==j) |
|  | { |
|  | continue loop1 |
|  | } |
|  | } |
|  | } |
|  |  |
|  | // 0 0 |
|  | // 1 0 |
|  | // 1 1 |
|  | // 2 0 |
|  | // 2 1 |
|  | // 2 2 |
|  | // 3 0 |
|  | // 3 1 |
|  | // 3 2 |
|  | // 3 3 |
|  | // 4 0 |
|  | // 4 1 |
|  | // 4 2 |
|  | // 4 3 |
|  | // 4 4 |

20. Rest parameters Syntax

|  |
| --- |
| //The rest parameter syntax allows a function to accept an indefinite number of arguments as an array,it like variadic function in C. |
|  | function abc(...args) |
|  | { |
|  | console.log(args) |
|  | } |
|  | abc(1) //[1] |
|  | abc(1,2) //[1,2] |
|  | abc(1,2,3) //[1,2,3] |
|  | abc(1,2,3,4)//[1,2,3,4[ |