Javascript Arrays

Road Map

1.Arrays
-What is an Array?
-Array Syntax [] , new Array().
-Replacing Array Values.
-Different data types in Array.
-Array Methods:
    length()
    pop(),push()
    Shift(),unshift(),
    splice()
    split()
    replace()
    indexOf() 0,-1 includes() true0r false

2.Looping in Array.
-Types of loop:- for,for of,while
-For loop.
-Loop Syntax->for(initialization of condition; testing condition; increment/decrement){ statement }
-Reading Arrays with for loop.
-Creating new based on the value of one array. Using push.
-Looping Backwards in an Array.
-For of loop in arrays.
-Comparison with for loop.
-Syntax of for loop.
-Looping through an Array to get each value.
-Entries method in for of loop.
-While loop

3.Destructuring Array
-What is Destructuring?
-Destructuring Assignment in arrays and objects.
-Modern and Old Way of Destructuring.
-Destructuring Array.
-Destructuring Array in Objects.

-Nested Destructuring.
-Flat Method

4.Set
-What is a set?
-Set syntax.
-Set Methods.
-Spread Operator
-Removing Duplicates from an array

Arrays

-An array is a list of items that stores more than one value.
-Instead of storing one value in our variable we can create an array.
-An array is created with [] notation or new Array() notation, values go into the brackets.
-Arrays can hold different data types also.

Syntax

[]
new Array()

Example

let friends= [‘Ian’,`Sharon`,`Lorem`];
let myFriends = new Array(‘Ian’,`Sharon`,`Lorem`);
Output:- [‘Ian’,`Sharon`,`Lorem`]
Output:- [‘Ian’,`Sharon`,`Lorem`];

Replacing array values

-We can replace the values of our array by calling the variable with its index value.
-The index we call is the array value we want to replace and we set the value to be replaced.

let myFriends = [‘Ian’,`Sharon`,`Lorem`];
myFriends[0]=`Sharon`;
Output:-[`Sharon`,`Sharon`,`Lorem`];

Array Methods

Slice()

-It extracts a piece of an array into a new array.
-Slice is immutant,It does not mutate original original array

let letters = [`a`,`b`,`c`,`d`,`e`];
letters.slice(2) - [`c`,`d`,`e`]
letter.slice(2,3) - [`c`,`d`]
letter.slice() - [`a`,`b`,`c`,`d`,`e`] -When no index set it returns duplicate.
console.log(letters) -All original arrays will be returned.

splice()

-It extracts a piece of an array into a new array.
-Slice is mutant, It mutates the original array.

let letters = [`a`,`b`,`c`,`d`,`e`];
letters.slice(2) - [`c`,`d`,`e`];
console.log(letters) - [`a`,`b`] -The arrays which were not extracted will be returned.

reverse()

-It reverses the order of elements in arrays.
-Reverse mutates the original array.

let letters = [`a`,`b`,`c`,`d`,`e`];
letters.reverse() -[`e`,`d`,`c`,`b`,`a`]
console.log(letters) - [`e`,`d`,`c`,`b`,`a`] -The original array order is overwritten.

concat()

-It joins two or more arrays together, Just like the spread operator.
-The method does not alter the original array.
let num1 = [`1`,`2`,`3`,`4`,`5`];
let num2 = [`6`,`7`,`8`,`9`,`10`];
let num3 = [`11`,`12`,`13`,`14`,`15`]
let allNum =num1.concat(num1,num2,num3);
Output:- [`1`,`2`,`3`,`4`,`5`,`6`,`7`,`8`,`9`,`10`,`11`,`12`,`13`,`14`,`15`]

An Easier alternative is using spread operator
Let allNum = [...num1,...num2,...num3]
Output:- [`1`,`2`,`3`,`4`,`5`,`6`,`7`,`8`,`9`,`10`,`11`,`12`,`13`,`14`,`15`]

join()
-It returns an array into a string.
-A separator can be specified , By default its (,)
let num1 = [`1`,`2`,`3`,`4`,`5`];
num.join(``)
Output:- 12345
num.join(`+`)
Output:- 1+2+3+4+5

length()
-Checks the length of Array
let items =[“cows”,”dog”,”goat”,”pigs”,”hen”]
Items.length
Output: 1

push()
-Adds element at end of array
let items =[“cows”,”dog”,”goat”,”pigs”,”hen”]
items.push(“buffalo”)
Output:- [“cows”,”dog”,”goat”,”pigs”,”hen”,”buffalo”]

unshift()

-Adds first element of an array but overwritten it

let items =[“cows”,”dog”,”goat”,”pigs”,”hen”]

items.unshift(“buffalo”)

Output:- [“buffalo”“cows”,”dog”,”goat”,”pigs”,”hen”]

Pop()

-Removes the last element from an array.

let items =[“cows”,”dog”,”goat”,”pigs”,”hen”];

items.pop()

Output:-[“cows”,”dog”,”goat”,”pigs”];

shift()

-Remove the first element of an array.

let items =[“cows”,”dog”,”goat”,”pigs”,”hen”]

items.shift()

Output:- [”dog”,”goat”,”pigs”,”hen”]

indexOf()

-It returns the indexOf the value specified from the array.

-The parameter indexOf is the value present in the array.

-If the value specified is not found it returns -1

-It searches from left to right.

let items =[“cows”,”dog”,”goat”,”pigs”,”hen”]

items.indexOf(`cows`)

Output:- 0

let items =[“cows”,”dog”,”goat”,”pigs”,”hen”]

items.indexOf(`buffalo`);

Output:- -1

Includes()

-It Returns either true or false if the value is in the array.

-True is returned if the value is present in the array.

-False is returned if the value is present in the array.

let items =[“cows”,”dog”,”goat”,”pigs”,”hen”];

items.include(`cow`)

items.include(`cows`)

Output:- true

Output:- false

Looping in Arrays.

-A loop allows us to repeat a specific operation a number of times without having to write that operation individually.

-Instead of logging the array items one by one by using its index number, A loop can be used to print them one by one.

Without a loop.

let scores = [22, 54, 76, 92, 43, 33]

scores[0]

scores[1]

scores[2]

scores[3]

scores[4]

scores[5]

Output:- 22, 54, 76, 92, 43, 33

With loop

let scores = [22, 54, 76, 92, 43, 33]

for(let score of scores)

Output:- 22, 54, 76, 92, 43, 33

Loops

-These are some of the loops.

1.for loop

2.for of loop

3.while loop

4.forEach

For loop in array

-For loop, loops through a code a number of times.

-It keeps running if condition is true.

-It has 3 expressions which is as follows in this syntax.

Syntax

for(initialization of condition; testing condition; increment/decrement){ statement }

Initialization of condition.

-Firstly we initialize the loop with a variable and value which is an integer ,and this marks where our loop starts.

Testing Condition.

-Here is the the stopping condition, It determine when the loop has to stop.

-Here our condition is tested and if it is true our loop body is executed.

Increment/Decrement.

-This increases the counter otherwise our code will remain at 1 forever.

Example

-Here our loop which is initialized to start at 0.

-The condition is set to less than 10,Where it will stop.

-An increment of i=i+1 is set which mean it will increase by one.

```
for(let i = 0 ; i<10; i++) console.log(`Hello there${i}`)}
```

Output:- Hello there is printed 9 times.

```
const jonas = [`Jonas`,`Schmedinarm`,2037-1931,`Teacher`,true]
for(let i = 0; i<jonas.length;i++){ console.log(jonas[i])}
```

Output:-`Jonas`,`Schmedinarm`,106,`Teacher`,true

Looping backwards in Array.

-We can loop from backwards in the array where our last values will be printed first in that order.

-Our initialization expression will be defined with length of array minus one.

-We then set our condition if its >=1.

let array = [`Emmanuel`,`John`,`Koech`,`Pogba`,`Chris`]
for(let i = array.length-1;i>=0;i–){console.log(array[0])}
Output:-[`Chris`,`Pogba`,`Koech`,`John`,`Emmanuel`]

For of Loop in Array

-Normally we used the for loop to iterate through an array,which is the old way.
-It was introduced in ES6 and it's a new and easier way to loop through an Array.

Previously,Using for loop

-Return with the index number
let arr = [10,10,10,11,13]
for(let i = 0; i<arr.length;i++){ console.log(arr[i]) }

Modern Javascript ,Using for of loop.

-For of loop is the new modern way of looping through arrays.
-It loops through an array and it can return the index of an array using entries() method.
-Destructuring is applied if we want both the index and each value from our array,Note this followed by entries method.

Syntax

let array = []
for(let value of array){output} - Will return each value only from the array.
for(let value of array.entries())- Will return eachvalue and its index starting from 0.
for(let [i,value] of array.entries())-Will return each value and its index starting from 0.

Examples

let numbers = [10,10,10,11,13]
for(let number of numbers)console.log(number)
Output:- 10 10 10 11 13

let numbers = [10,10,10,11,13]
for(let number of numbers.entries())console.log(number)

Output:- 0 1 1 10 2 10 3 11 4 13

While Loop in array

-It loops through a block of code as long as a specified condition is True.

-Unlike the for loop the condition variable is initialized outside the loop.

Syntax

while(condition){ code block to be executed; counter}

Example

Here our condition is Initialized.

```
let i = 10;
Our condition is tested here
while(i<=10){
    console.log(`Hello${i}`);
    Increment happens here
    i++ ;
}
```

forEach Loop in array and set

-For each is an iterative method used in an array,It calls a provided callback function.

-The forEach has a function call back,which is an anonymous function .

-The array comes first followed by the forEach method.

-Inside the method is the anonymous function, Arrow functions can be used.

-The parameters of the function carry the (current element, index , entire array).

-The parameters method in sets does not carry an index.

Syntax of forEach loop

arrayName.forEach(anonymous function)
arrayName.forEach((value,index,array)=>) **-In Arrays**
arrayName.forEach((value,_,array)=>) -**In sets**

Example

```
let cars = [`Benz`,`Audi`,`Toyota`,`Bentley`,`Volgswagen`];
cars.forEach((car ,index,array)=> `${index}, ${car}`)
```

Output:- 0 Benz 1 Audi 2 Toyota 3 Bentley 4 Volkswagen

Example

```
let currenciesUnique = new Set([`USD`,`GBP`,`USD`,`EUR`,`EUR`]);
currenciesUnique.forEach((value,_,set)=>value)
```

Output:- `Benz` `Audi` `Toyota` `Bentley` `Volkswagen`

NB:-You cannot break out in the ForEach Loop, No Continue and Break Statement.

Data Transformation methods in Javascript

-These are methods that help us transform our data.

-They are 3

1.Map Method

2.Filter Method

3.Reduce Method.

Map Method

-It iterates over an array and returns a new array.

-Its main difference from for loop is that it creates a new array.

-In forEach for us to create new array we have to use the push method to an empty created array variable.

Syntax of Map

```
arrayName.map(function(value,index,array){})
```

Example Using Map

```
const cars =  [`Benz`,`Audi`,`Toyota`,`Bentley`,`Volgswagen`];
cars.map((car,index)=>{
        console.log(car)
})
```

Output:-[`Benz`,`Audi`,`Toyota`,`Bentley`,`Volgswagen`];

Example Using forEach

```
const cars =  [`Benz`,`Audi`,`Toyota`,`Bentley`,`Volgswagen`];
const newCarArray = []
cars.map((car,index)=>{
        newCarArray.push(car)
})
```

-Here we had to use the Push method to push the car to the new car array method.

Filter Method

-It is an iterative method in the array that creates a new array of elements that pass a test provided by a function.

-It does not change the original array.

Syntax

arrayName.filter(function(value,index,arr),thisValue)

Example

```
const movement = [200,450,-400, 3000,-650,70,1300];
const deposit = movement.filter((mov,index)=>{
        return mov>0;
        })
console.log(deposit)
```

OutPut:- [200, 450, 3000, 70, 1300]

Reduce Method

-It is used when you have array values and you want to add them all up.

-It loops through the array hence it is an iterator.

-It has 4 parameters (accumulator,value,index,array).

-The accumulator is the sum of all array elements.

-The value is each iterator in the array.

Syntax.

arrayName.reduce(function(accumulator,value,index,array){0})

Example.

```
const movements = [200,450,-400, 3000,-650,70,1300];
const balance = movement.reduce(total,value,index)=> total+value,0
balance()
```

Output:-

Find Method.

-It is an iterative method that returns the first value of an Array when a certain condition is satisfied.

-It works the same as the filter method but returns the first value of the array.

-The difference between find and filter is that filter returns a new array of all the conditions satisfied from the array whereas find returns only the first value from the array which the condition satisfies.No array is returned in find.

Syntax.

```
const arrs = [values];
const newArr = arrs.find(function(arr,index,array));
```

Example.

```
const movements = [200,450,-400, 3000,-650,70,1300];
movements.find(movement=>movement<-1)
```

Output:-400

Example

```
const account1 = {
        owner:`Jonas Schmrdtmann`,
        movement:[200, 450. -400, 3000 , -650, -130, 70 ,1300],
        interestRate: 1.2;
        pin:1111
}
```

```
const account2 = {
        owner:`Jessica Davis`,
        movement:[5000, 3400. -150, -790 , -650, -3210, 1000 ,8500, 30],
        interestRate: 1.5;
        pin:222
}

const accounts = [account1,account2];
accounts.find((acc=>acc.owner === `Jessica Davis` ));
```

Output:-{

```
        owner:`Jessica Davis`,
        movement:[5000, 3400. -150, -790 , -650, -3210, 1000 ,8500, 30],
        interestRate: 1.5;
        pin:222
        }
```

Output was the second array

-Here the FindIndex loops through the accounts array.
-AfterWards we check for the account owner if it satisfies the conditions which is set.
- 1 is the output since the owner is in the second array which is account 2.

FindIndex

-It returns the index of the first element of an array that satisfies the testing function.
-If no element satisfies the testing condition -1 is returned.

Syntax

```
const arrs =[]
arrs.findIndex(function(arr,index,array))
```

Example

```
let arrs = [200,450,-400, 3000,-650,70,1300];
arrs.findIndex(arr=>arr);
```

Output- 0

Example

```
const account1 = {
        owner:`Jonas Schmrdtmann`,
        movement:[200, 450. -400, 3000 , -650, -130, 70 ,1300],
        interestRate: 1.2;
        pin:1111
}
const account2 = {
        owner:`Jessica Davis`,
        movement:[5000, 3400. -150, -790 , -650, -3210, 1000 ,8500, 30],
        interestRate: 1.5;
        pin:222
}
const accounts = [account1,account2];
accounts.findindex((acc=>acc.owner === `Jessica Davis` ));
```

Output = 1

-Here the FindIndex loops through the accounts array.
-AfterWards we check for the account owner.
-1 is the output since the owner is in the second array which is account 2.

Destructuring Arrays.

-Destructuring is the process of breaking an array down and extracting only what is needed.

-In destructuring the order that variables are declared is important.

-If we only wanted a car and a suv we skip truck but put a comma.

Destructuring Assignment

-It makes it possible to unpack values from arrays or Properties from objects into distinct variables.

-They are [],{}.Respectively in arrays and objects.

Old Way Of Destructuring Array

```
let vehicles = [`Mustang`,`f-150`,`expedition`];
let car = vehicle[0];
let truck = vehicle[1];
let suv = vehicle[2];
```

Output:- `Mustang`,`f-150`,`expedition`

Modern Way of Destructuring Array(ES6)

```
let vehicles = [`Mustang`,`f-150`,`expedition`];
let[car,truck,suv]=vehicles;
```

-Skip Truck

```
let[car,,suv]=vehicles;
```

Destructuring Arrays if its in an Object

```
const restaurant ={
        name:`Classico Italiano`,
        location:`Via Angelo Tavaanti 23,Firenze,Italy`,
        categories:[`Italian`,`Pizzeria`,`Vegetarian`,`Organic`],
        starterMenu:[`Focaccia`,`Bruschetta`,`Garlic Bread`,`Caprese Salad`],
        mainMenu:[`Pizza`,`Pasta`,`Risoto`],
```

```
        openingHours:{
                        thu:{
                         open:12,
                         close:22,
                        },
                        fri:{
                          open:11,
                          close:23,
                        },
                        sat:{
                            open:11,
                            close:23,
                        }

                        }
}
```

```
const [first,second,third]=restaurant.categories
```

Destructuring

-Changing Positions

```
const [first,second,third]=restaurant.categories
[first,second,third]=[third,second,first]
```

Nested Destructuring

-Arrays can be inside an array,these are nested arrays that need to be destructured.

-Here All of the Arrays will be printed

```
let categories = [`Italian`,`Pizzeria`,[`Vegetarian`,`Organic`]];
let[first,second,third,]=categories
```

Output:-`Italian`,`Pizzeria`,[`Vegetarian`,`Organic`]

-For each item in the array to be printed we nest our variable,when destructuring.

let categories = [`Italian`,`Pizzeria`,[`Vegetarian`,`Organic`]];
let[first,second,[third,four]]=categories
Output:-`Italian`,`Pizzeria`,`Vegetarian`,`Organic`

Note:- To Avoid Nested Destructuring We can use the flat method.

Sets

-Sets are collections of unique values,They are almost like arrays.
-We define a set using new set().
-It can hold any data type.
-Each value can occur only once in a set.
-Sets are iterables
-Main use case of set is to remove duplicate values from an array.
-Note sets do not have indexes.

Syntax

let setName = new Set();
let setName = new Set([]);

Set Methods

-These Methods are added after the set name.

add() -Add new value to the set, New value goes inside the bracket.
delete()-Removes value, Value we want to delete goes inside the bracket.
has()-Checks if the value exists in the set,Value we want to check goes in the bracket.
clear()-It removes all the values in the set.
size - Checks the size of set

Removing Duplicate Values from arrays

-Main use case is to remove duplicate values from an array.
-The spread operator copies the existing array into another array.
let arr = [`Pineapple`,`Apple`,`Oranges`,`Avocado`,`Pears`,Pineapple`,`Apple`]

let arrNonDup = new Set([...arr])
Output:{`Pineapple`,`Apple`,`Oranges`,`Avocado`,`Pears`}

Example

let fruits = new
Sets([`Pineapple`,`Apple`,`Oranges`,`Avocado`,`Pears`,Pineapple`,`Apple`,`Oranges`])
Output:{`Pineapple`,`Apple`,`Oranges`,`Avocado`,`Pears`}

let fruits = new
Sets(`Pineapple`,`Apple`,`Oranges`,`Avocado`,`Pears`,Pineapple`,`Apple`,`Oranges`)
Output:{`P`i`n`e`a`p`p`le`}