Frontend Development

Array Concept:

An array is a special variable, which can hold more than one value at a time. The Array object let's you store multiple values in a single variable. In Java Script array is a linear and homogeneous data structure which is used for to store data in a linear and sequential way. The homogeneous means in an array all the elements data type are same type of data type.

Array Creation:

```js
let ar1 = [1,2,3,4,5];
console.log(ar1);

let ar2 = ["MAKS", "AKS", "SKS"];
console.log(ar2);
```

```
PS E:\HTML_CSS_JS> node demo.js
[ 1, 2, 3, 4, 5 ]
[ 'MAKS', 'AKS', 'SKS' ]
PS E:\HTML_CSS_JS>
```

```js
let ar1 = [12.5, 25.33, 5.41, 11.2];
console.log(ar1);

let ar2 = [true, false, true, false];
console.log(ar2);
```

```
PS E:\HTML_CSS_JS> node demo.js
[ 12.5, 25.33, 5.41, 11.2 ]
[ true, false, true, false ]
PS E:\HTML_CSS_JS>
```

```js
let ar1 = [12, -15, 2.53, 'A', "MAKS",false];
console.log(ar1,"\n");

let ar2 = [
    {name: "MAKS"},
    {name: "SKS"},
    {name: "AKS"}
];
console.log(ar2,"\n");

let ar3 = [
    [1,2,3],
    {name:"JavaScript"},
    12
];
console.log(ar3);
```

```
PS E:\HTML_CSS_JS> node demo.js
[ 12, -15, 2.53, 'A', 'MAKS', false ]

[ { name: 'MAKS' }, { name: 'SKS' }, { name: 'AKS' } ]

[ [ 1, 2, 3 ], { name: 'JavaScript' }, 12 ]
PS E:\HTML_CSS_JS>
```

```js

let ar1 = new Array();
let size = 5;
for(let i = 0; i < size; i++){
    ar1[i] = prompt("");
}
console.log(ar1);
```

```
PS E:\HTML_CSS_JS> node demo.js
12
MAKS
true
25.66
-666
[ '12', 'MAKS', 'true', '25.66', '-666' ]
PS E:\HTML_CSS_JS>
```

Different Techniques Of Accessing Elements & Print Elements:

demo.js > ar

```js
let ar = [1,2,3,4,5];
for(let i = 0; i < ar.length; i++){
    console.log(ar[i]);
}
console.log('-----------------------');
for(let x in ar){
    console.log(ar[x]);
}
console.log('-----------------------');
for(let z of ar){
    console.log(z);
}
```

```
PS E:\HTML_CSS_JS> node demo.js
1
2
3
4
5
-----------------------
1
2
3
4
5
-----------------------
1
2
3
4
5
PS E:\HTML_CSS_JS>
```

demo.js > ...

```js
let ar = [1,2,3,4,5,6];
ar.forEach((i)=>{
    console.log(i);
})
console.log("---------------------");
let ar1 = ar.forEach((j)=>{
    console.log(j * 2);
})
```

```
PS E:\HTML_CSS_JS> node demo.js
1
2
3
4
5
6
----------------------
2
4
6
8
10
12
PS E:\HTML_CSS_JS>
```

demo.js > ar2 > ar.forEach() callback

```js
let ar = [
    {name: "MAKS", age: 22},
    {name: "SKS", age: 15},
    {name: "AKS", age: 29}
];
console.log(ar);
console.log("---------------------------");
ar.forEach((i)=>{
    console.log(i.age);
})
let ar1 = ar.forEach((j)=>{
    console.log(j.name);
})
console.log("---------------------------");
let ar2 = ar.forEach((j)=>{
    console.log(j);
})
```

```
PS E:\HTML_CSS_JS> node demo.js
[
  { name: 'MAKS', age: 22 },
  { name: 'SKS', age: 15 },
  { name: 'AKS', age: 29 }
]
-----------------------------
22
15
29
MAKS
SKS
AKS
-----------------------------
{ name: 'MAKS', age: 22 }
{ name: 'SKS', age: 15 }
{ name: 'AKS', age: 29 }
PS E:\HTML_CSS_JS>
```

Array Methods:

at():

at() this method is used for to display the given index number's element & count starts from 0.

```
let ar = [1,2,3,4,5,6,7,8];
console.log(ar.at(3));
```

```
PS E:\HTML_CSS_JS> node demo.js
4
PS E:\HTML_CSS_JS>
```

concat():

this method concatenates(joins) two or more arrays. The concat() method returns a new array, containing the joined arrays. The concat() method does not change the existing arrays.

```
let ar = [1,2,3,4];
let ar1 = ["a","b","c"];
console.log(ar.concat(ar1));
```

```
PS E:\HTML_CSS_JS> node demo.js
[
  1,  2,  3,  4,
  'a', 'b', 'c'
]
PS E:\HTML_CSS_JS>
```

Constructor:

this property returns the function that created the array prototype.

```
let ar = [1,2,3,4];
console.log(ar.constructor);
```

```
PS E:\HTML_CSS_JS> node demo.js
[Function: Array]
PS E:\HTML_CSS_JS>
```

indexOf():

this method returns the first index position of a specified value. this function returns -1 if the value is not found. This method starts at a specified index and searches from left to right.

```
let ar = [1,2,3,4,5,6,7,8,9,0];
console.log(ar.indexOf(5));
```

```
PS E:\HTML_CSS_JS> node demo.js
4
PS E:\HTML_CSS_JS>
```

IsArray():

this function is used for to check whether the array is a valid array or not.

```
let ar = [1,2,3,4,5,6,7,8];
console.log(Array.isArray(ar));
```

```
PS E:\HTML_CSS_JS> node demo.j
s
true
PS E:\HTML_CSS_JS>
```

Join():

this method returns an array as a String. this function does not change the original array. any separator can be specified. The default is comma.

```
demo.js > ...
let ar = [1,2,3];
console.log(ar.join("!"));
```

```
PS E:\HTML_CSS_JS> node demo.js
1!2!3
PS E:\HTML_CSS_JS>
```

Keys():

this function return an array iterator object with the keys of an array. this function method does not change the original array.

```
demo.js > ...
let ar = [1,2,3,4,5,6];
for(let x of ar.keys()){
console.log(x,":",ar[x]);
}
```

```
PS E:\HTML_CSS_JS> node demo.js
0 : 1
1 : 2
2 : 3
3 : 4
4 : 5
5 : 6
PS E:\HTML_CSS_JS>
```

lastIndexOf():

this method returns the lastIndex(position) of a specified value. the lastIndexOf() method returns -1 if the value is not found. The lastIndexOf() starts at a specified index and searches from right to left. By default the search starts at the last element and ends at the first. Negative start values counts from the last element(but still searches from right to left).

```
demo.js
demo.js > ...
let ar = [1,2,3,2,4,2];
console.log(ar.lastIndexOf(2));
```

```
powershell
PS E:\HTML_CSS_JS> node demo.js
5
PS E:\HTML_CSS_JS>
```

length: this is used for to find the size of array.

```
demo.js
demo.js > ...
let ar = [1,2,3,2,4,2];
console.log(ar.length);
```

```
powershell
PS E:\HTML_CSS_JS> node demo.js
6
PS E:\HTML_CSS_JS>
```

Push():

this method adds new items to the end of an array. the push() method changes the length of the array. the push() method returns the new length. If you print then count start from 1.

```
demo.js > ...
let ar = [1,2,3];
ar.push(20);
ar.push(40);
console.log(ar);
```

```
PS E:\HTML_CSS_JS> node demo.js
[ 1, 2, 3, 20, 40 ]
PS E:\HTML_CSS_JS>
```

Pop():

this method removes the last element of an array. the pop method changes the original array. the pop() method returns the removed element.

```
demo.js > ...
let ar = [1,2,3,20,40];
ar.pop(20);
ar.pop(40);
console.log(ar);
```

```
PS E:\HTML_CSS_JS> node demo.js
[ 1, 2, 3 ]
PS E:\HTML_CSS_JS>
```

Reverse():

this method reverse the order of the elements in an array. the reverse() method overwrites the original array.

```
demo.js > ...
let ar = [1,2,3,20,40];
console.log(ar);
console.log(ar.reverse());
```

```
PS E:\HTML_CSS_JS> node demo.js
[ 1, 2, 3, 20, 40 ]
[ 40, 20, 3, 2, 1 ]
PS E:\HTML_CSS_JS>
```

Slice():

this method returns selected elements in an array. as a new array. the slice method selects from a given start up to a (not inclusive) given end. The slice() method does not change the original array. print from 0 to n – 1

```
demo.js > ...
let ar = [1,2,3,4,5,6,7,8];
console.log(ar);
console.log(ar.slice(1,5));
```

```
PS E:\HTML_CSS_JS> node demo.js
[
  1, 2, 3, 4,
  5, 6, 7, 8
]
[ 2, 3, 4, 5 ]
PS E:\HTML_CSS_JS>
```

Shift():

this method removes the first item of an array. the shift() method changes the original array. the shift() method returns the shifted element.

```
demo.js > ...
let ar = [1,2,3,4,5,6,7,8];
console.log(ar.shift());
console.log(ar);
console.log(ar.shift());
console.log(ar);
```

```
PS E:\HTML_CSS_JS> node demo.js
1
[
  2, 3, 4, 5,
  6, 7, 8
]
2
[ 3, 4, 5, 6, 7, 8 ]
PS E:\HTML_CSS_JS>
```

valueOf(): this method returns the itself. This method does not change the original array.

```
demo.js > ...
let ar = [1,2,3,4,5];
console.log(ar.valueOf());
```

```
PS E:\HTML_CSS_JS> node demo.js
[ 1, 2, 3, 4, 5 ]
PS E:\HTML_CSS_JS>
```

Unshift():

this method adds new elements to the beginning of an array. the unshift() method overwrites the original array. if you print the method the array count starts from 1.

```
demo.js > ar
let ar = [1,2,3,4,5];
ar.unshift(7);
ar.unshift(8);
console.log(ar);
```

```
PS E:\HTML_CSS_JS> node demo.js
[
  8, 7, 1, 2,
  3, 4, 5
]
PS E:\HTML_CSS_JS>
```

Sort():

the method sort the element of array. this method overwrites the original array. the sort() sorts the elements as strings in alphabetical and ascending order.

```
demo.js
demo.js > ...
let ar = [50,20,70,10,60,30,40,100,90,80];
console.log(ar.sort((a,b)=>a-b));
console.log(ar.sort((a,b)=>a-b).reverse());
```

```
powershell
PS E:\HTML_CSS_JS> node demo.js
[
  10, 20, 30, 40, 50,
  60, 70, 80, 90, 100
]
[
  100, 90, 80, 70, 60,
   50, 40, 30, 20, 10
]
PS E:\HTML_CSS_JS>
```

Splice():

this method adds and/or removes array elements. The splice() method overwrites the original array.

```
demo.js
demo.js > ...
let ar = [1,2,3,4,5,6,7,8,9,0];
ar.splice(1,0,10,20);
console.log(ar);
ar.splice(5,2);
console.log(ar);
```

```
powershell
PS E:\HTML_CSS_JS> node demo.js
[
  1, 10, 20, 2, 3,
  4,  5,  6, 7, 8,
  9,  0
]
[
  1, 10, 20, 2, 3,
  6,  7,  8, 9, 0
]
PS E:\HTML_CSS_JS>
```

Some():

this function checks if any array elements pass a test provides as a callback function. this function method executes the callback function once for each array element. The some() method returns true of the function returns true for one of the array elements. This function returns false. If the function returns false for all of the array elements. The some() method does not execute the function for empty array elements. The some() method does not change the original array.

```
demo.js > fun
let ar = [50,20,70,10,60,30,40,100,90,80];
console.log(ar.some(fun));
function fun(ar){
    return ar < 50;
}
```

```
powershell
PS E:\HTML_CSS_JS> node demo.js
true
[
  50, 20, 70, 10, 60,
  30, 40, 100, 90, 80
]
PS E:\HTML_CSS_JS>
```

Map():

this function creates a new array from calling a function for every array element. This function does not execute the function for empty element. Map() doesn't change the original array.

```
demo.js > ar.map() callback
let ar = [1,2,3,4,5,6,7,8,9,0];
ar.map((i)=>{
    if(i <= 5)
        console.log(i);
})
```

```
powershell
PS E:\HTML_CSS_JS> node demo.js
1
2
3
4
5
0
PS E:\HTML_CSS_JS>
```

```
demo.js > ...
let ar = [
    {name:"MAKS", age: 23},
    {name:"AKS", age: 29},
    {name:"SKS", age: 15}
];
ar.map((i)=>{
    console.log(i.age+" "+typeof(i.age));
})
```

```
powershell
PS E:\HTML_CSS_JS> node demo.js
23 number
29 number
15 number
PS E:\HTML_CSS_JS>
```

Filter():

demo.js > ar.filter() callback

```
let ar = [1,2,3,4,5,6,7,8,9,0];
ar.filter((i)=>{
    if(i % 2 === 0) console.log(i);
})
```

powershell

```
PS E:\HTML_CSS_JS> node demo.js
2
4
6
8
0
PS E:\HTML_CSS_JS>
```

demo.js > ar.filter() callback

```
let ar = [
    {age:15},
    {age:11},
    {age:18},
    {age:14},
    {age:20}
];
ar.filter((i)=>{
    if(i.age >= 15) console.log(i);
})
```

powershell

```
PS E:\HTML_CSS_JS> node demo.js
{ age: 15 }
{ age: 18 }
{ age: 20 }
PS E:\HTML_CSS_JS>
```

Reduce():

demo.js > ar.reduce() callback

```
let ar = [1,2,3,4,5,6,7,8,9];
ar.reduce((acc,cur)=>{
    if(cur >= 5) console.log(cur);
},[])
```

powershell

```
PS E:\HTML_CSS_JS> node demo.js
5
6
7
8
9
PS E:\HTML_CSS_JS>
```

demo.js > ar.reduce() callback

```
let ar = [
    {name:"MAKS",age:22},
    {name:"SKS",age:15},
    {name:"LKS",age:20},
    {name:"GKS",age:14},
    {name:"MKS",age:11},
    {name:"OKS",age:12},
];
ar.reduce((acc,cur)=>{
    if(cur.age >= 14) console.log(cur);
},[])
```

powershell

```
PS E:\HTML_CSS_JS> node demo.js
{ name: 'MAKS', age: 22 }
{ name: 'SKS', age: 15 }
{ name: 'LKS', age: 20 }
{ name: 'GKS', age: 14 }
PS E:\HTML_CSS_JS>
```

Array In React:

App.js

ReactZH > src > App.js > default

```
Click here to ask Blackbox to help you code faster
import React from 'react';
import './App.css';
import Array from './Array';
import Box from './Components/Box';
function App() {
  return (
    <>
      <h1 className='h1'>List Of Members</h1>
      <div className='LB'>
        <Box
          name = {Array[0].name} age = {Array[0].age} dob = {Array[0].dob}
        />
        <Box
          name = {Array[1].name} age = {Array[1].age} dob = {Array[1].dob}
        />
        <Box
          name = {Array[2].name} age = {Array[2].age} dob = {Array[2].dob}
        />
        <Box
          name = {Array[3].name} age = {Array[3].age} dob = {Array[3].dob}
        />
      </div>
    </>
  );
}
export default App;
```

Array.js

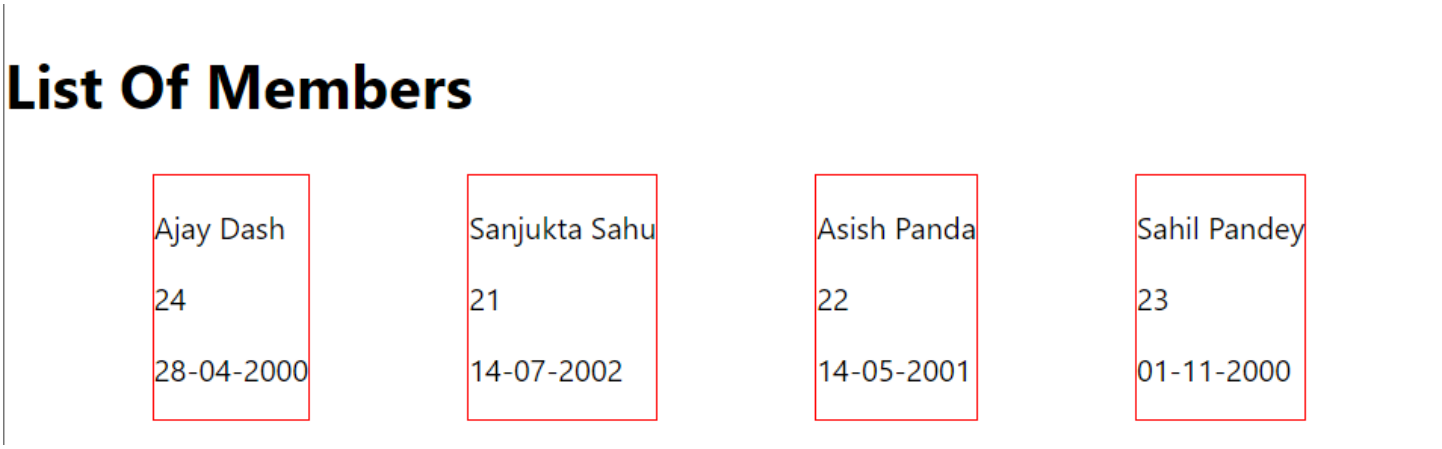
ReactZH > src > Array.js > Array > dob

```
Click here to ask Blackbox to
let Array = [
  {
    name:"Ajay Dash",
    age: 24,
    dob: "28-04-2000"
  },
  {
    name:"Sanjukta Sahu",
    age: 21,
    dob: "14-07-2002"
  },
  {
    name:"Asish Panda",
    age: 22,
    dob: "14-05-2001"
  },
  {
    name:"Sahil Pandey",
    age: 23,
    dob: "01-11-2000"
  }
];
export default Array;
```

Box.js

ReactZH > src > Components > Box.js > Box

```
Click here to ask Blackbox to help you code faster
import React from "react";
const Box = (props)=>{
  return(
    <div className="box">
      <p className="name">{props.name}</p>
      <p className="age">{props.age}</p>
      <p className="dob">{props.dob}</p>
    </div>
  );
}
export default Box;
```



Map method needs a unique key. Keys are necessary to improve performance of your react app.

Array Map method in react:

App.js

ReactZH > src > App.js > App > Array.map() callback

```
  return (
    <>
      <h1 className='h1'>List Of Members</h1>
      <div className='LB'>
        {Array.map((i, index)=>{
          console.log(index);
          return(
            <Box
             key = {index}
             name = {i.name}
             age = {i.age}
             dob = {i.dob}
            />
          );
        })}
      </div>
    </>
  );
}
export default App;
```

Box.js

ReactZH > src > Components > Box.js > Box

```
Click here to ask Blackbox to help you code faster
import React from "react";
import './Box.css';
const Box = (props)=>{
  return(
    <div className="box">
      <p className="name">{props.name}</p>
      <p className="age">{props.age}</p>
      <p className="dob">{props.dob}</p>
    </div>
  );
}
export default Box;
```

Array.js

ReactZH > src > Array.js > default

```
Click here to ask Blackbox to help you code faster
let Array = [
  {
    name:"Ajay Dash",
    age: 24,
    dob: "28-04-2000"
  },
  {
    name:"Sanjukta Sahu",
    age: 21,
    dob: "14-07-2002"
  },
  {
    name:"Asish Panda",
    age: 22,
    dob: "14-05-2001"
  },
  {
    name:"Sahil Pandey",
    age: 23,
    dob: "01-11-2000"
  }
];
export default Array;
```

List Of Members

Ajay Dash 24 28-04-2000	Sanjukta Sahu 21 14-07-2002	Asish Panda 22 14-05-2001	Sahil Pandey 23 01-11-2000

Why Keys? Keys help react identify which items have changed faded/removed/re-ordered to give a unique identity to every element inside the array, a key is required.