

# React Map

A map is a data collection type where data is stored in the form of pairs. It contains a unique key. The value stored in the map must be mapped to the key. We cannot store a duplicate pair in the map(). It is because of the uniqueness of each stored key. It is mainly used for fast searching and looking up data.

In React, the `map` method is used to traverse and display a list of similar objects of a component. A map is not the feature of React. Instead, it is the standard JavaScript function that could be called on any array. The `map()` method creates a new array by calling a provided function on every element in the calling array.

## Example

In the given example, the `map()` function takes an array of numbers and double their values. We assign the new array returned by `map()` to the variable `doubleValue` and log it.

1. `var numbers = [1, 2, 3, 4, 5];`
2. `const doubleValue = numbers.map((number)=>{`
3. `return (number * 2);`
4. `});`
5. `console.log(doubleValue);`

## In React, the `map()` method is used for:

1. Traversing the list element.

### Example

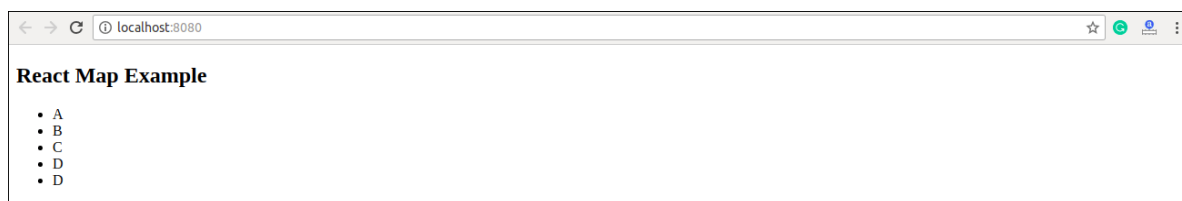
1. `import React from 'react';`
2. `import ReactDOM from 'react-dom';`
- 3.
4. `function NameList(props) {`
5. `const myLists = props.myLists;`
6. `const listItems = myLists.map((myList) =>`
7. `<li>{myList}</li>`
8. `);`
9. `return (`
10. `<div>`

```

11.     <h2>React Map Example</h2>
12.     <ul>{listItems}</ul>
13. </div>
14. );
15.}
16. const myLists = ['A', 'B', 'C', 'D', 'D'];
17. ReactDOM.render(
18. <NameList myLists={myLists} />,
19. document.getElementById('app')
20.);
21. export default App;

```

## Output



2. Traversing the list element with keys.

## Example

```

1. import React from 'react';
2. import ReactDOM from 'react-dom';
3.
4. function ListItem(props) {
5.   return <li>{props.value}</li>;
6. }
7.
8. function NumberList(props) {
9.   const numbers = props.numbers;
10.  const listItems = numbers.map((number) =>
11.    <ListItem key={number.toString()}
12.      value={number} />
13.  );
14.  return (
15.    <div>
16.      <h2>React Map Example</h2>

```

```
17.     <ul> {listItems} </ul>
18. </div>
19. );
20. }
21.
22. const numbers = [1, 2, 3, 4, 5];
23. ReactDOM.render(
24.   <NumberList numbers={numbers} />,
25.   document.getElementById('app')
26. );
27. export default App;
```

## Output



# React Table

A table is an arrangement which organizes information into rows and columns. It is used to store and display data in a structured format.

The react-table is a lightweight, fast, fully customizable (JSX, templates, state, styles, callbacks), and extendable Datagrid built for React. It is fully controllable via optional props and callbacks.

## Features

1. It is lightweight at 11kb (and only need 2kb more for styles).
2. It is fully customizable (JSX, templates, state, styles, callbacks).
3. It is fully controllable via optional props and callbacks.
4. It has client-side & Server-side pagination.
5. It has filters.
6. Pivoting & Aggregation
7. Minimal design & easily themeable

## Installation

Let us create a **React app** using the following command.

1. `$ npx create-react-app myreactapp`

Next, we need to install **react-table**. We can install react-table via npm command, which is given below.

1. `$ npm install react-table`

Once, we have installed react-table, we need to **import** the react-table into the react component. To do this, open the **src/App.js** file and add the following snippet.

1. **import** ReactTable from "react-table";

Let us assume we have data which needs to be rendered using react-table.

1. `const data = [{`
2.     `name: 'Ayaan',`
3.     `age: 26`

```
4.     },{
5.     name: 'Ahana',
6.     age: 22
7.     },{
8.     name: 'Peter',
9.     age: 40
10.    },{
11.    name: 'Virat',
12.    age: 30
13.    },{
14.    name: 'Rohit',
15.    age: 32
16.    },{
17.    name: 'Dhoni',
18.    age: 37
19.    }]
```

Along with data, we also need to specify the **column info** with **column attributes**.

```
1.  const columns = [{
2.    Header: 'Name',
3.    accessor: 'name'
4.  },{
5.    Header: 'Age',
6.    accessor: 'age'
7.  }]
```

Inside the render method, we need to bind this data with react-table and then returns the react-table.

```
1.  return (
2.    <div>
3.      <ReactTable
4.        data={data}
5.        columns={columns}
6.        defaultPageSize = {2}
7.        pageSizeOptions = {[2,4, 6]}
8.      />
```

9. </div>

10.)

Now, our **src/App.js** file looks like as below.

1. **import** React, { Component } from 'react';

2. **import** ReactTable from "react-table";

3. **import** "react-table/react-table.css";

4.

5. **class** App **extends** Component {

6.   render() {

7.     **const** data = [{

8.       name: 'Ayaan',

9.       age: 26

10.     },{

11.       name: 'Ahana',

12.       age: 22

13.     },{

14.       name: 'Peter',

15.       age: 40

16.     },{

17.       name: 'Virat',

18.       age: 30

19.     },{

20.       name: 'Rohit',

21.       age: 32

22.     },{

23.       name: 'Dhoni',

24.       age: 37

25.     }]

26.   **const** columns = [{

27.      Header: 'Name',

28.      accessor: 'name'

29.    },{

30.      Header: 'Age',

31.      accessor: 'age'

32.    }]

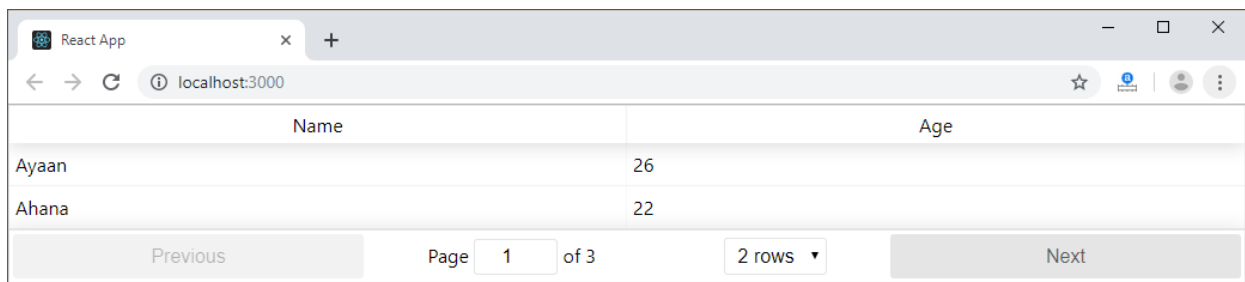
```

33. return (
34.   <div>
35.     <ReactTable
36.       data={data}
37.       columns={columns}
38.       defaultPageSize = {2}
39.       pageSizeOptions = {[2,4, 6]}
40.     />
41.   </div>
42. )
43. }
44. }
45. export default App;

```

## Output

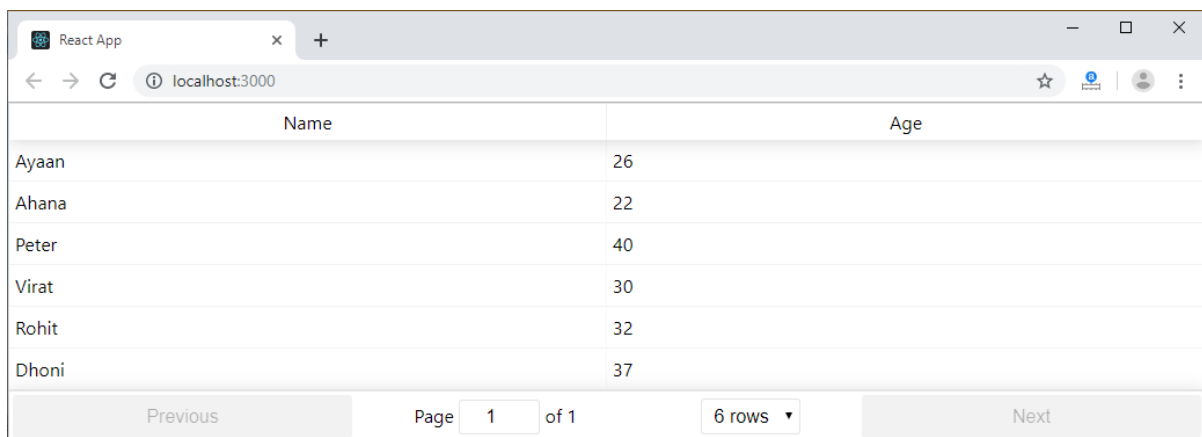
When we execute the React app, we will get the output as below.



Name	Age
Ayaan	26
Ahana	22

Previous Page 1 of 3 2 rows Next

Now, change the rows dropdown menu, we will get the output as below.



Name	Age
Ayaan	26
Ahana	22
Peter	40
Virat	30
Rohit	32
Dhoni	37

Previous Page 1 of 1 6 rows Next