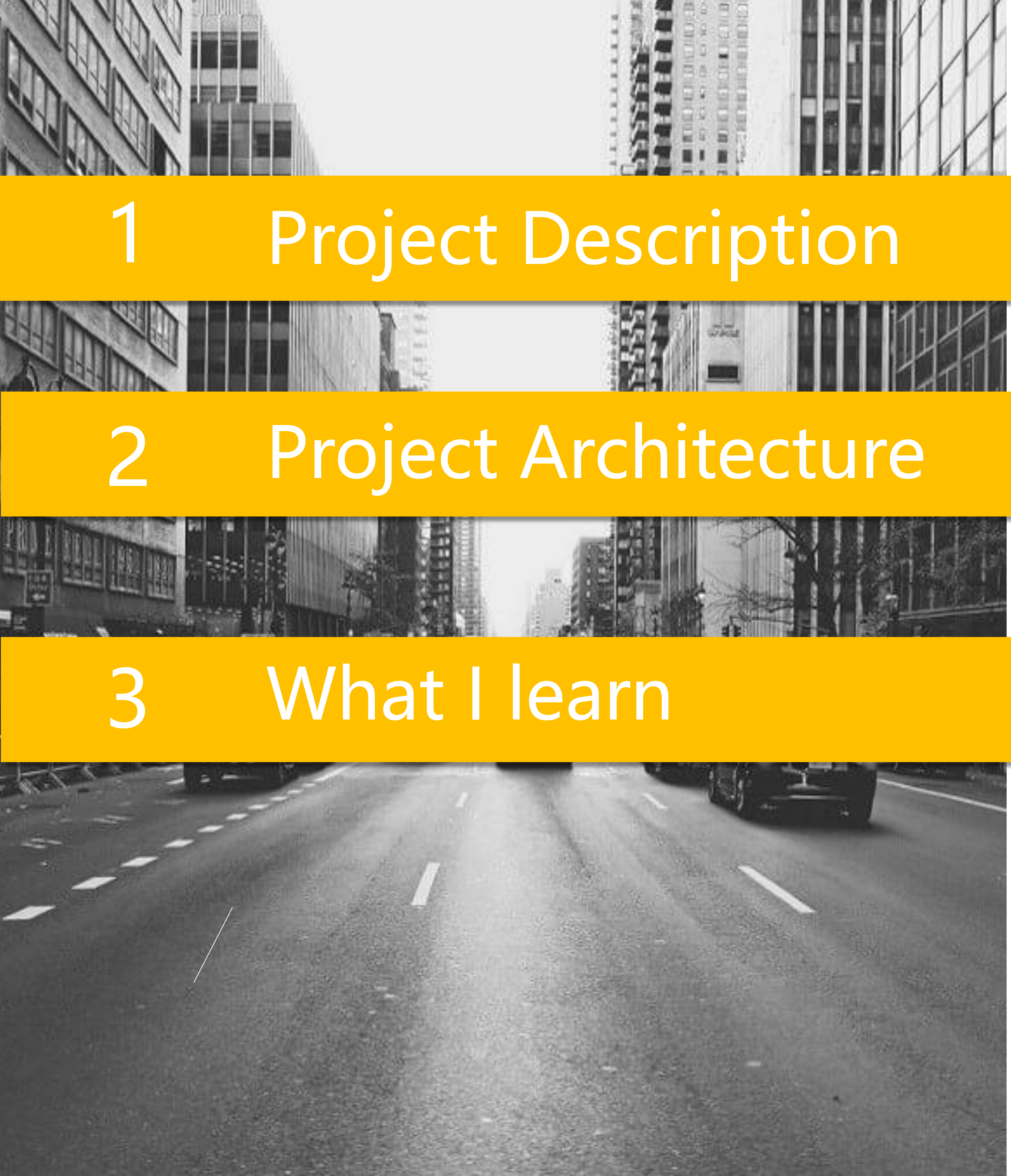
WordCloud App

Made by Jane Shan



WordCloud App

Project Description

Wordcloud App - a simple Web Spider

1. Use Axios and cheerio for web scraping
2. Use WordFrequenter to split string and count words
3. Use React.js for front-end and react-d3-cloud for visualization.
4. Use Node.js for back-end.

02

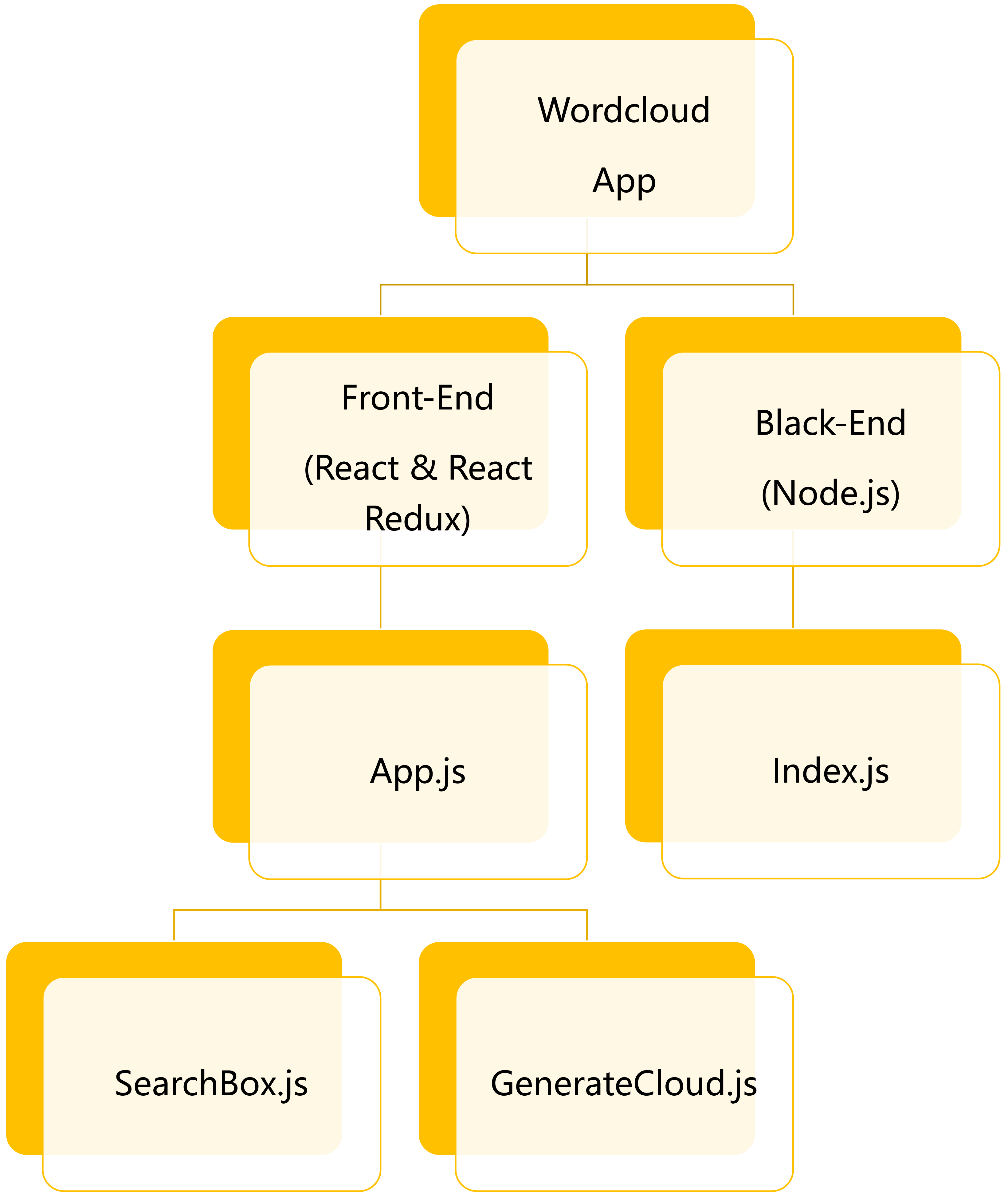
Project Architecture

1.File Structure & Project Dependencies

2. Code Logic

3. Code Snippet

File Structure & Project Dependencies



```
Front-End
"dependencies": {
"antd": "^3.20.1",
"axios": "^0.19.0",
"react": "^16.8.6",
"react-d3-cloud": "^0.7.0",
"react-dom": "^16.8.6",
"react-redux": "^7.1.0",
"react-scripts": "3.0.1",
"redux": "^4.0.1"
}
```

```
Back-End
"dependencies": {
"axios": "^0.19.0",
"body-parser": "^1.19.0",
"cheerio": "^1.0.0-rc.3",
"cors": "^2.8.5",
"express": "^4.17.1",
"underscore": "^1.9.1",
"wordfrequenter": "^1.0.0"
}
```

Code Logic

Client Side

1. User input URL in search box
2. When user click search button, Client Side send **POST** request with **URL** as params to server side by **axios**

Server Side

1. When Server Side get Post request, it will pass URL to **axiosGet(url)** function

axiosGet(url) function do following thing:

1. Send **GET** request to the given URL by **axios**.
2. Pass the returned **html document** to **cheerio**, cheerio get the web content by css selectors
3. **wordfrequenter (wf)** will do words splitting and counting
4. Return the **lists** generated by wf to Client Side

1. Client Side receive the Lists then pass the lists to action creator

addListToStore()

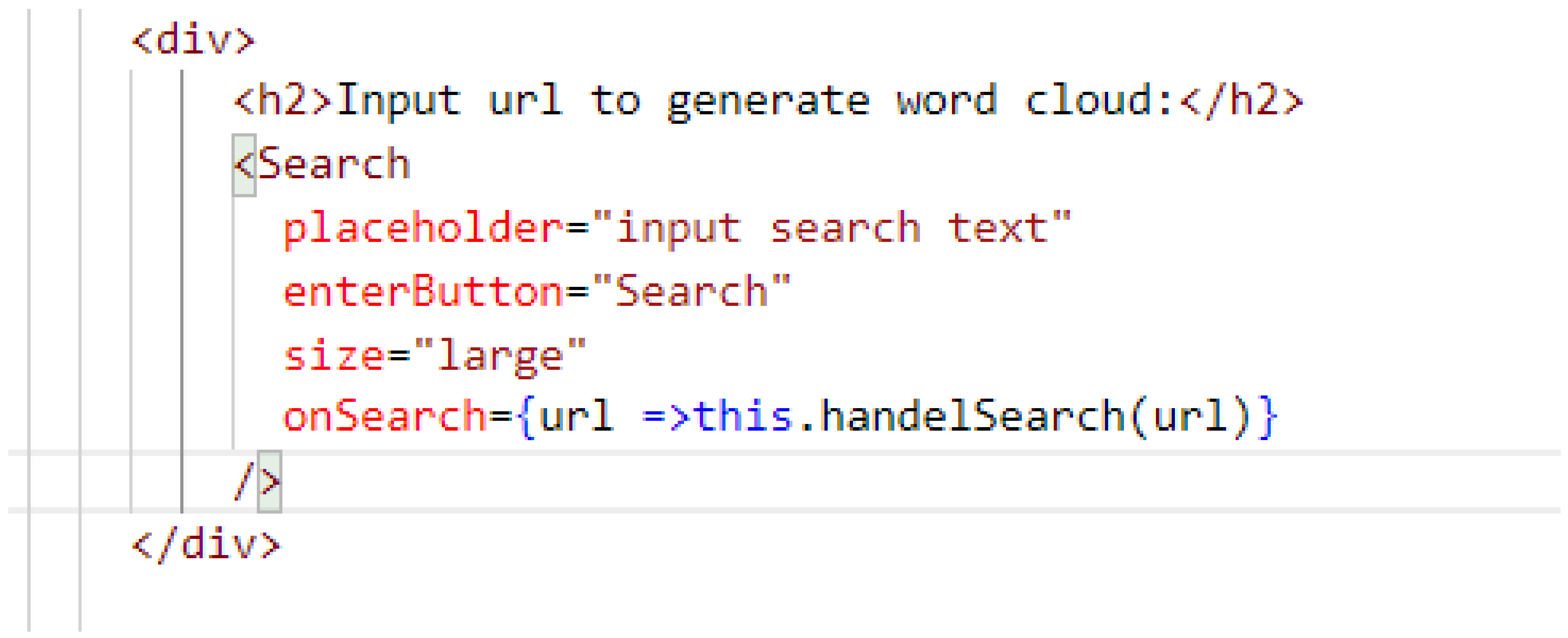
2. Then store’ s state will be replaced by the lists

3. Generate Word cloud

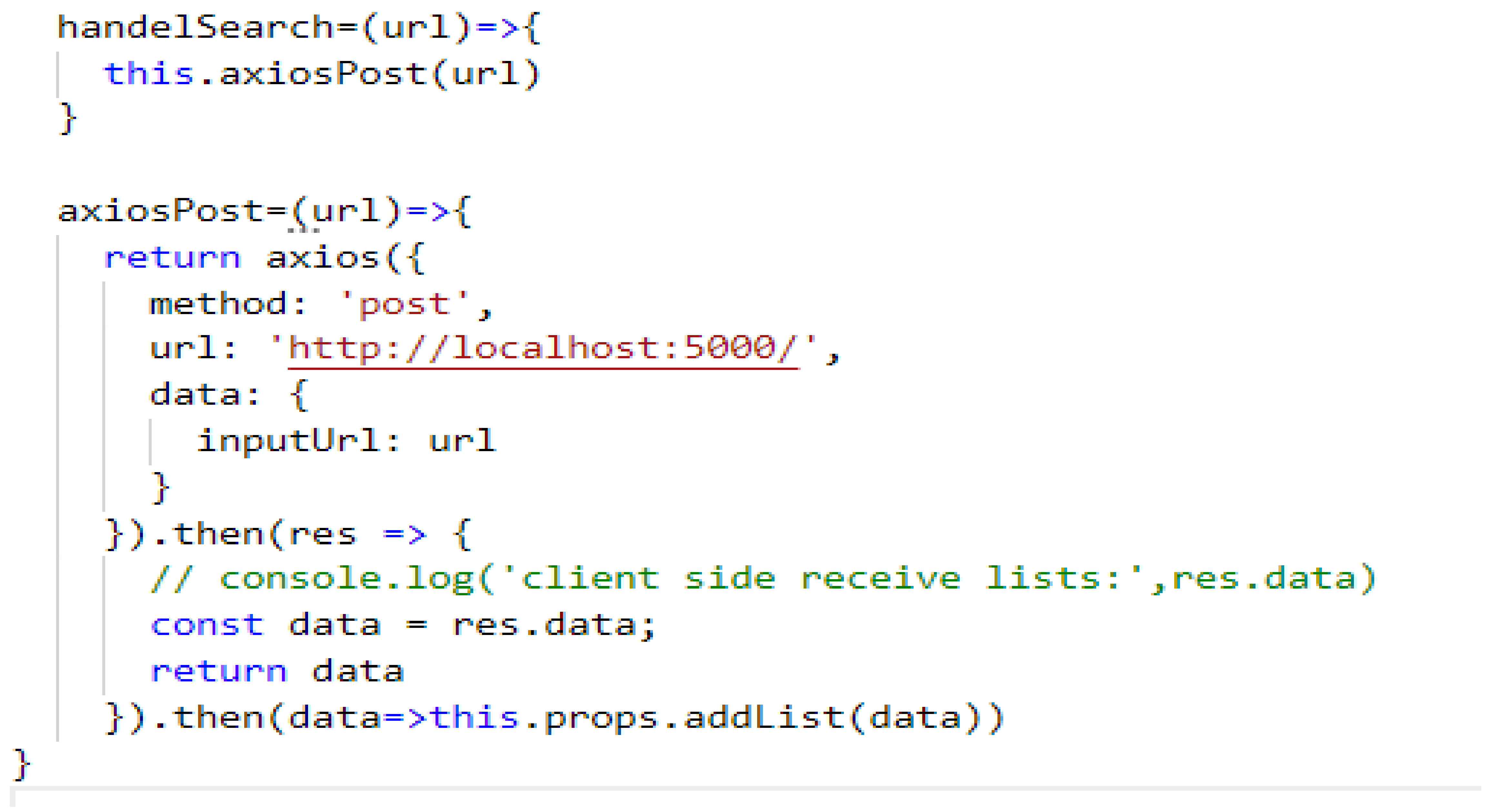
Client Side-SearchBar

1.User input URL in search bar

2. When user click search button, Client Side send **POST** request with **URL** as params to server side by **axios**



```
<div>
    <h2>Input url to generate word cloud:</h2>
    <Search
      placeholder="input search text"
      enterButton="Search"
      size="large"
      onSearch={url =>this.handelSearch(url)}
    />
</div>
```



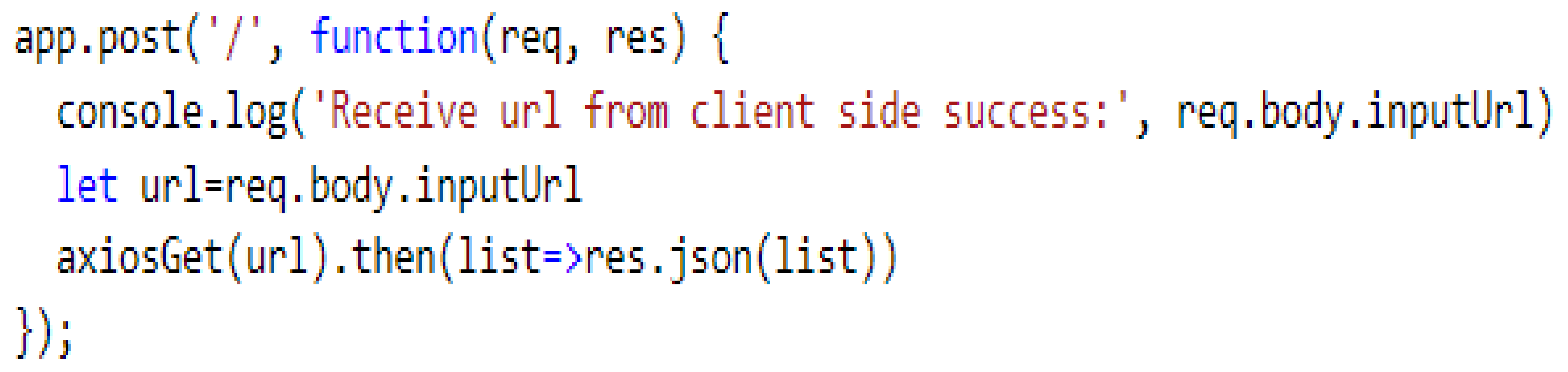
```
handelSearch=(url)=>{
  this.axiosPost(url)
}

axiosPost=(url)=>{
  return axios({
    method: 'post',
    url: 'http://localhost:5000/',
    data: {
      inputUrl: url
    }
  }).then(res => {
    // console.log('client side receive lists:',res.data)
    const data = res.data;
    return data
  }).then(data=>this.props.addList(data))
}
```

Server Side

1. When Server Side get Post request, it will pass URL to axiosGet(url) function

PHOTO FROM PIXABAY

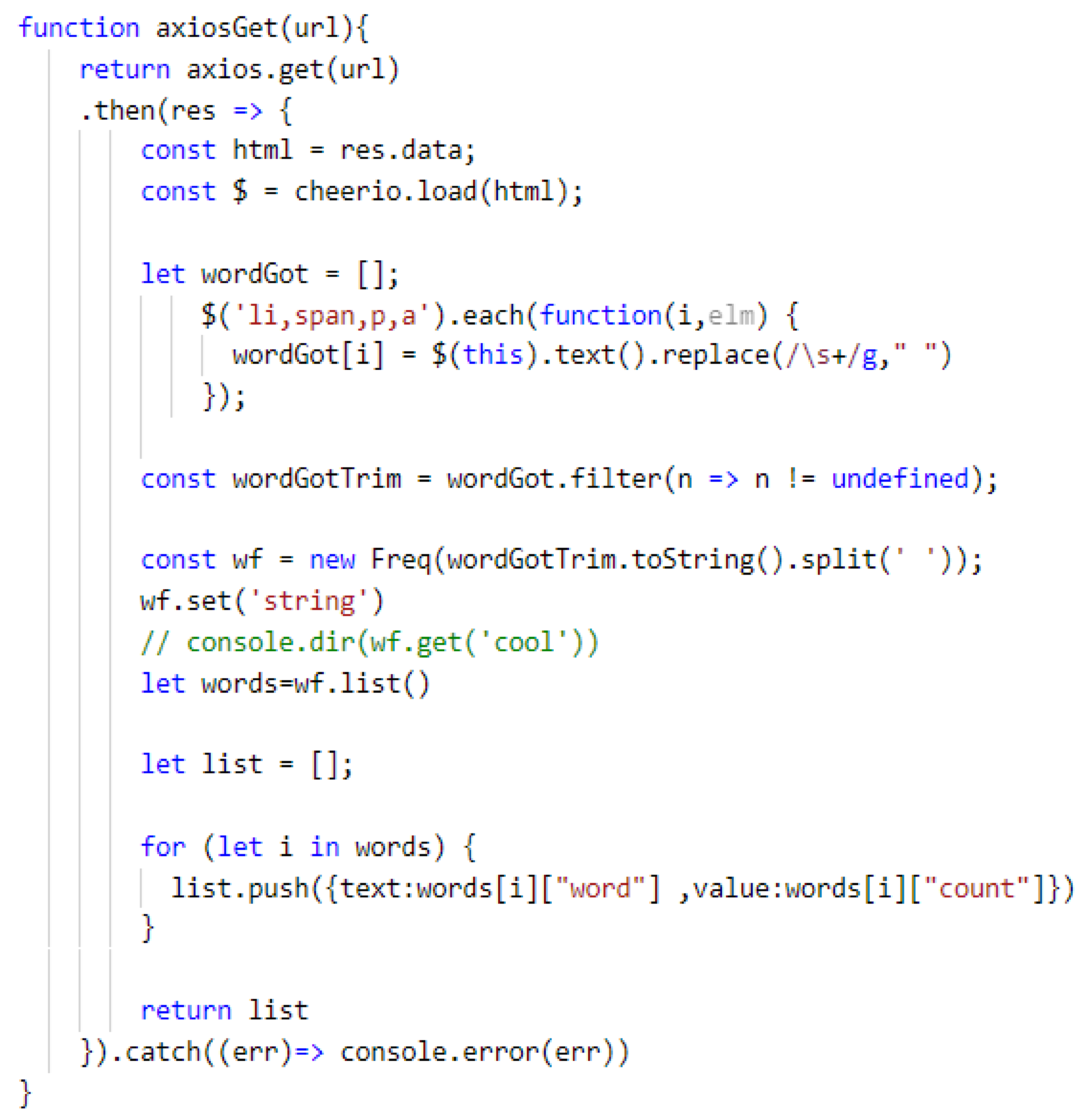


```
app.post('/', function(req, res) {
  console.log('Receive url from client side success:', req.body.inputUrl)
  let url=req.body.inputUrl
  axiosGet(url).then(list=>res.json(list))
});
```

Server Side

2. When user click search button, Client Side send **POST** request with **URL** as params to server side by **axios**

PHOTO FROM PIXABAY



```
function axiosGet(url){
    return axios.get(url)
    .then(res => {
        const html = res.data;
        const $ = cheerio.load(html);

        let wordGot = [];
            $('li,span,p,a').each(function(i,elm) {
              wordGot[i] = $(this).text().replace(/\s+/g," ")
            });

        const wordGotTrim = wordGot.filter(n => n != undefined);

        const wf = new Freq(wordGotTrim.toString().split(' '));
        wf.set('string')
        // console.dir(wf.get('cool'))
        let words=wf.list()

        let list = [];

        for (let i in words) {
          list.push({text:words[i]["word"] ,value:words[i]["count"]})
        }

        return list
    }).catch((err)=> console.error(err))
}
```

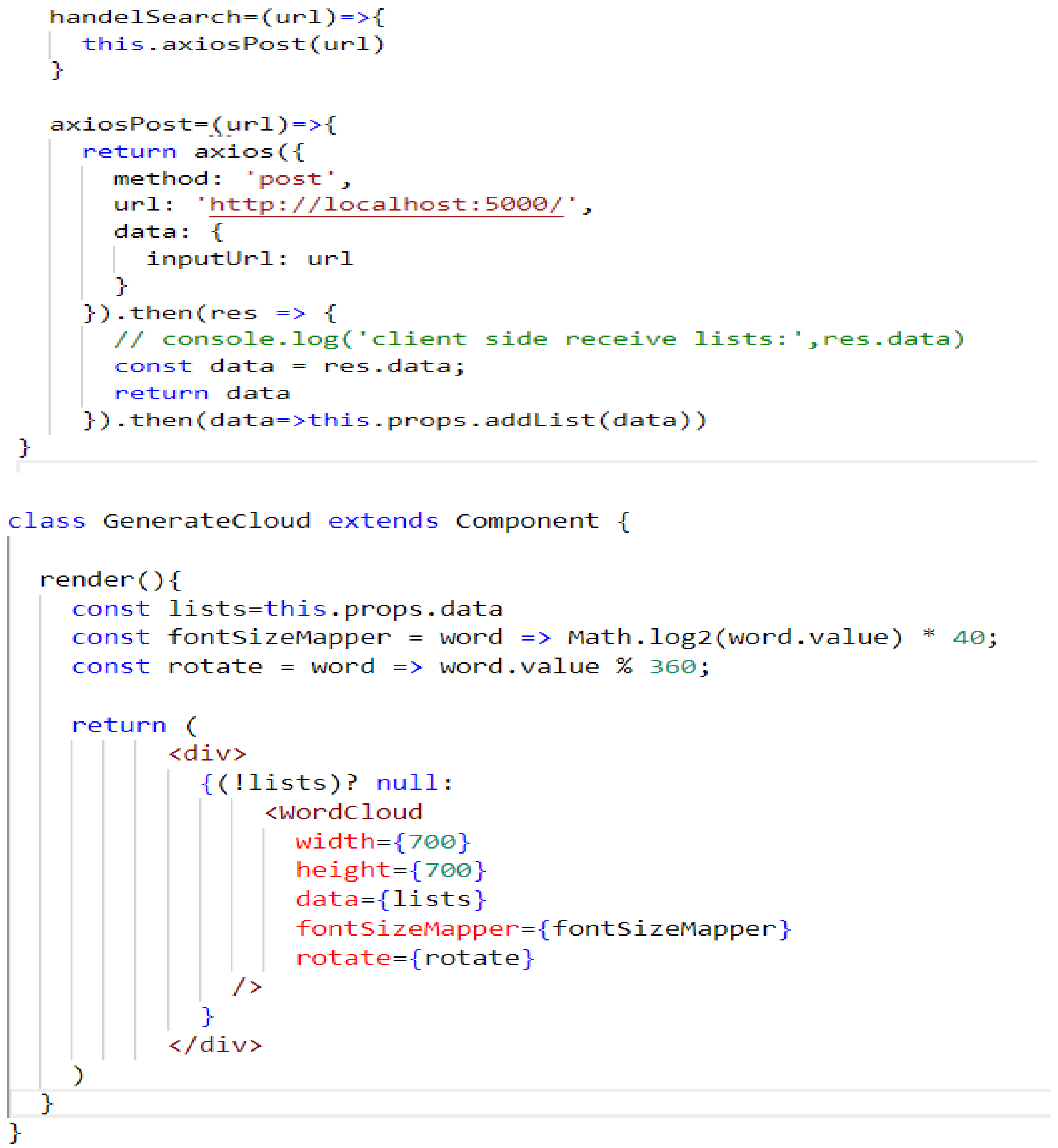
Client Side-GenerateCloud

1. Client Side receive the Lists then pass the lists to action creator

addListToStore()

2. Then store’ s state will be replaced by the lists

3. Generate the WordCloud



```
  handelSearch=(url)=>{
    this.axiosPost(url)
  }

  axiosPost=(url)=>{
    return axios({
      method: 'post',
      url: 'http://localhost:5000/',
      data: {
        inputUrl: url
      }
    }).then(res => {
      // console.log('client side receive lists:',res.data)
      const data = res.data;
      return data
    }).then(data=>this.props.addList(data))
}

class GenerateCloud extends Component {

  render(){
    const lists=this.props.data
    const fontSizeMapper = word => Math.log2(word.value) * 40;
    const rotate = word => word.value % 360;

    return (
          <div>
            {(!lists)? null:
              <WordCloud
                width={700}
                height={700}
                data={lists}
                fontSizeMapper={fontSizeMapper}
                rotate={rotate}
              />
            }
          </div>
    )
  }
}
```

THANKS FOR WATCHING