

# React

## List and Item

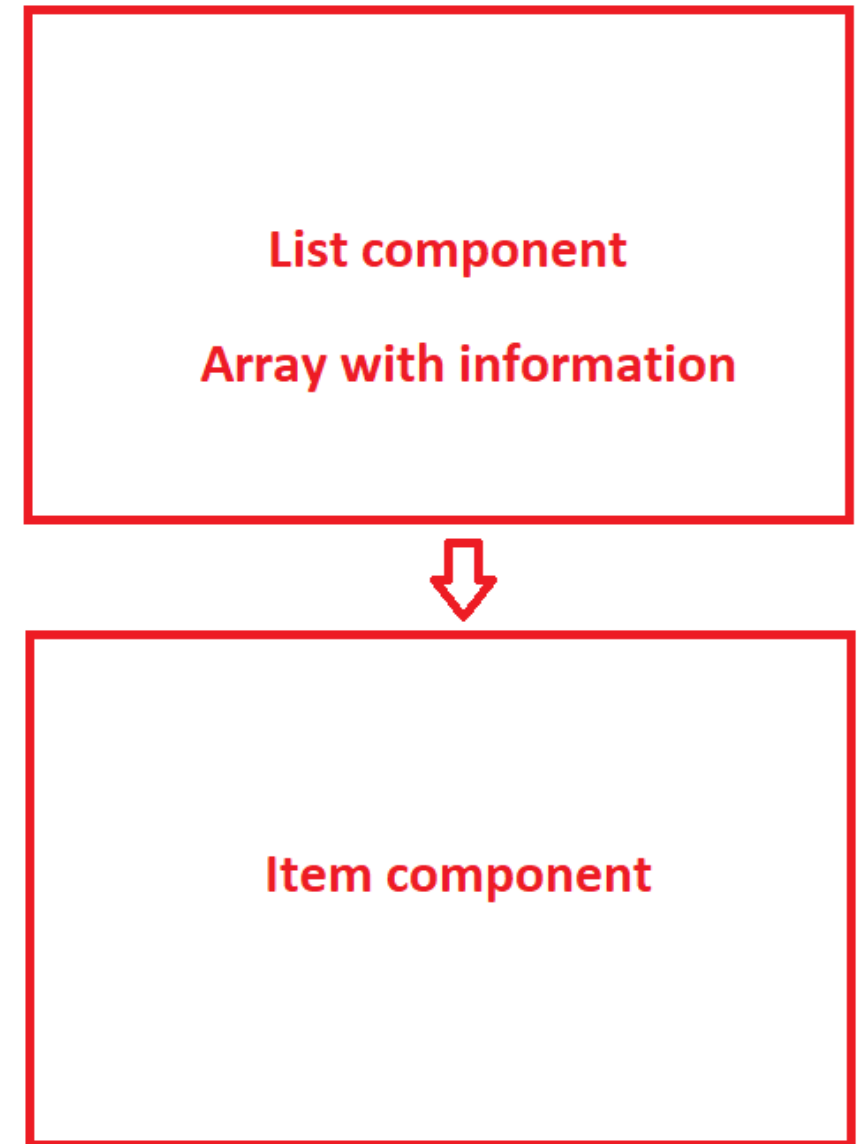
Rolando Gonzalez, 2022

# Content

- The List and Item components
- Key
- Code example
- Reference

# List and Item

- List and Item is about splitting the responsibility of an Item deciding what to show and how to show it, while the List generates x number of Item components.
- For example in a product page you will have a List component that has an array of objects. The List component uses a loop to generate x number of Items.



# The List component

- Contains an array with information.
- Generates Item components
  - Typically with ES6 .map function
- Passes down props to the Item components

# The Item component

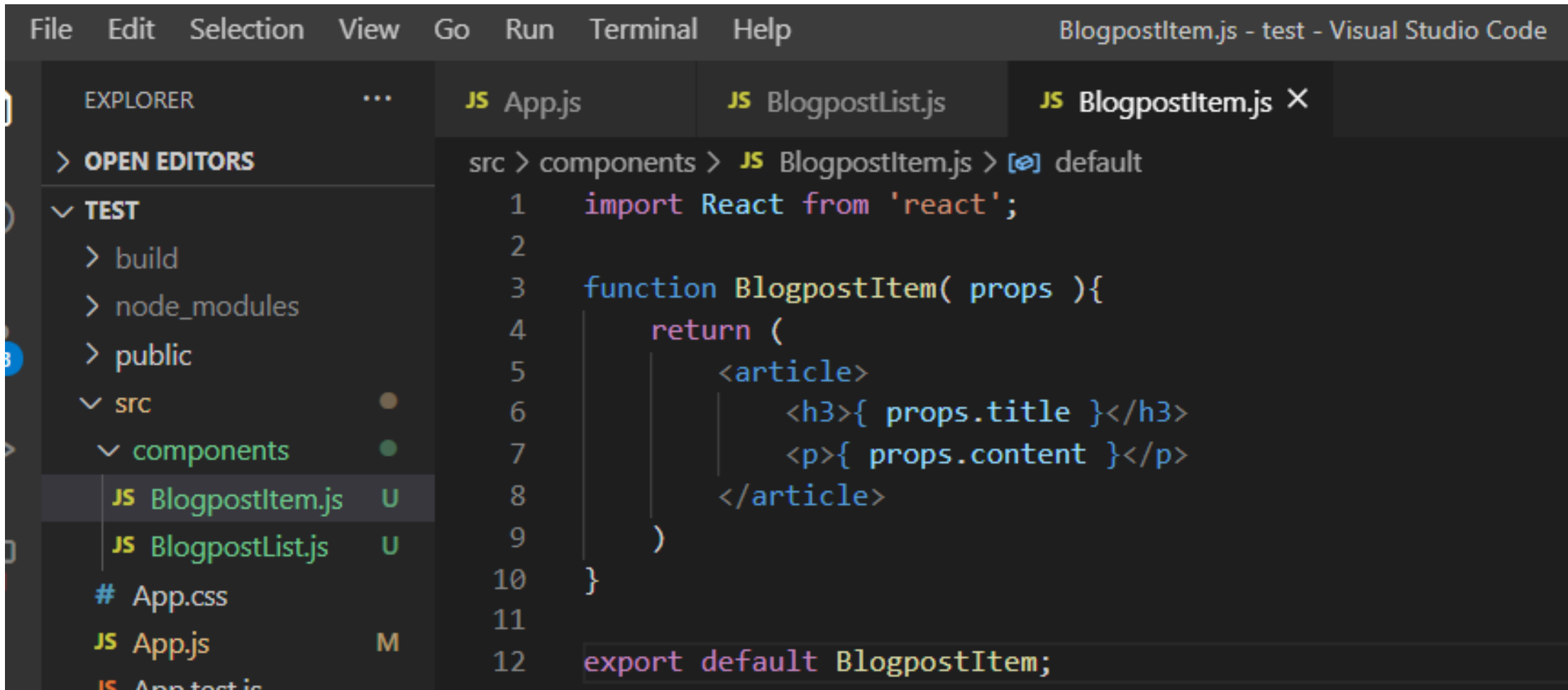
- The Item component is a component which is to be shown on the webpage.
- It will receive props from its List parent component

# Key

- When generating Item components in the List component you will need to give each Item component a unique Key.
- The unique Key is for enabling React in knowing exactly which Item component on the page to update when necessary

# Code example: Item

- The item is created for receiving props from List component.



The screenshot shows the Visual Studio Code interface with the 'BlogpostItem.js' file open. The Explorer sidebar on the left shows the project structure with 'src > components > BlogpostItem.js' selected. The main editor area displays the following code:

```
src > components > JS BlogpostItem.js > [🔍] default
1  import React from 'react';
2
3  function BlogpostItem( props ){
4      return (
5          <article>
6              <h3>{ props.title }</h3>
7              <p>{ props.content }</p>
8          </article>
9      )
10 }
11
12 export default BlogpostItem;
```

# Code example: List

- The List component will need some data source. In this example a function containing an array of objects is called when generating Item components.
- This array may also be received from App or through an Ajax call.

```
BlogpostList.js - test - Visual Studio Code

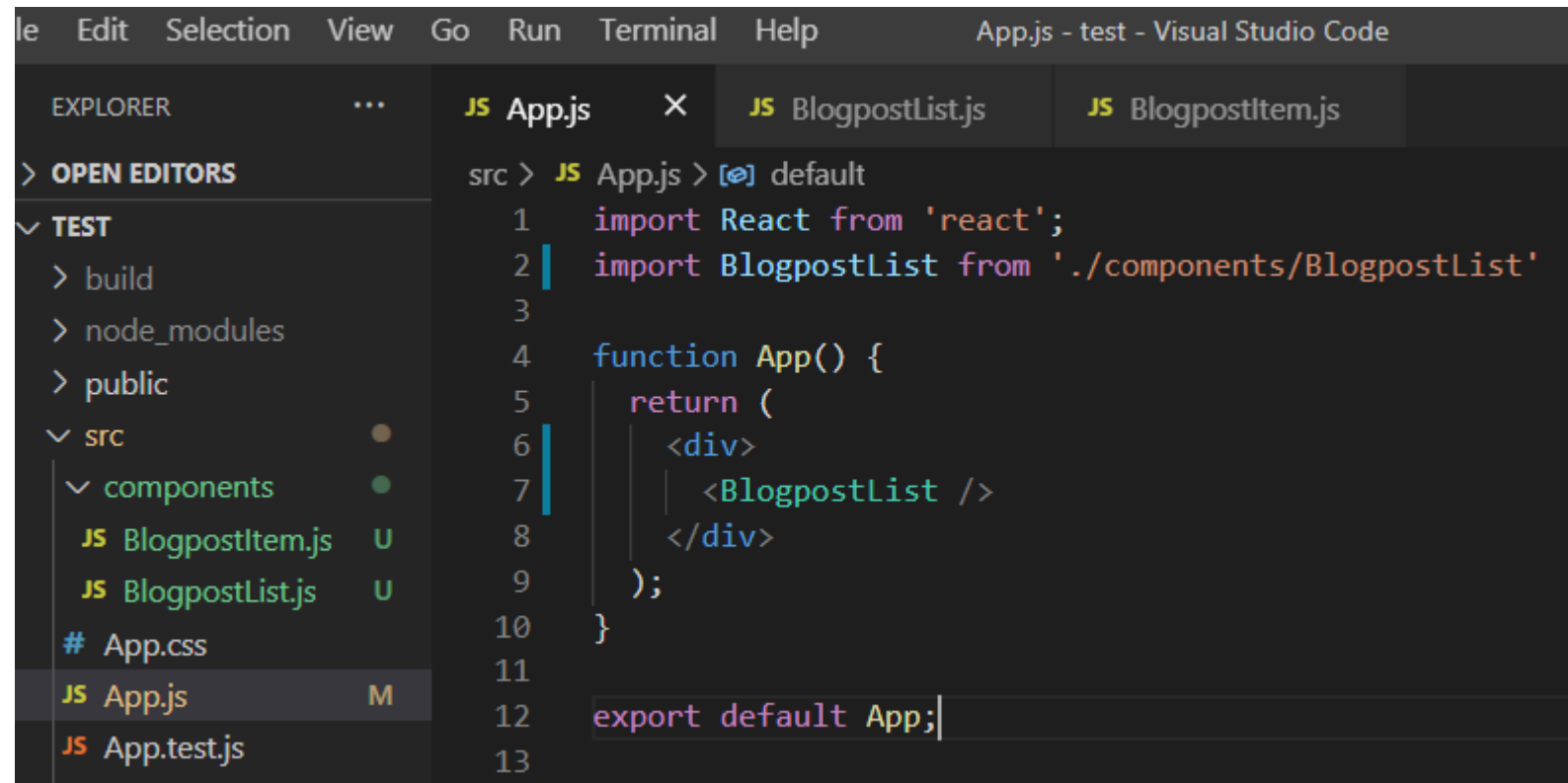
EXPLORER
  OPEN EDITORS
  TEST
    build
    node_modules
    public
    src
      components
        BlogpostItem.js
        BlogpostList.js
  App.css
  App.js
  App.test.js
  index.css
  index.js
  logo.svg
  serviceWorker.js
  setupTests.js
  .gitignore
  package.json
  package-lock.json
  README.md

src > components > BlogpostList.js > BlogpostList > generateBlogposts > map() callback
1  import React from 'react';
2  import BlogpostItem from './BlogpostItem';
3
4  function BlogpostList(){
5
6      function generateBlogposts(){
7          return getBlogposts().map( ( blogpost, i ) => {
8              return <BlogpostItem
9                  key={ "b" + i } title={ blogpost.title } content={ blogpost.content } />
10          })
11      }
12
13      return(
14          <section>
15              { generateBlogposts() }
16          </section>
17      )
18  }
19
20  export default BlogpostList;
21
22  function getBlogposts(){
23      const blogpostArray = [
24          {
25              title: "Nice weather",
26              content: "The weather is going to be very nice today."
27          },
28          {
29              title: "Bad weather",
30              content: "The weather is going to be very nice today."
31          }
32      ];
33      return blogpostArray;
34  }
```



# Code example App

- The App (or other appropriate component) imports the List component.



The screenshot shows the Visual Studio Code interface with the following details:

- Explorer Panel:** Displays a file tree with the following structure:
  - TEST
    - build
    - node\_modules
    - public
    - src
      - components
        - BlogpostItem.js (U)
        - BlogpostList.js (U)
      - App.css (#)
      - App.js (M)
      - App.test.js
- Editor Panel:** Shows the `App.js` file with the following code:

```
src > JS App.js > [default]
1  import React from 'react';
2  import BlogpostList from '../components/BlogpostList'
3
4  function App() {
5    return (
6      <div>
7        <BlogpostList />
8      </div>
9    );
10 }
11
12 export default App;
```
- Terminal Panel:** Empty.

# Reference

- Explains separation of responsibility:
  - <https://reactjs.org/docs/higher-order-components.html#use-hocs-for-cross-cutting-concerns>