Step-by-step process for finding and assembling components

written by Hong Le Nguyen 08.2024

Our goal

is finding out the components of UI app and their composition so that
the components are reusable,
the communication between them is less,
the re-rendering of the respective component (*) is less
(*) There are reasons to render a component: the initial render, the component (or one of its ancestors) state has been changed.

To ensure that the components do not arise by chance, we should follow a process. I outline *my (action-data centric) process* using an example the **"Book Review" UI app**.

1)
functionality description of "Book-Review" UI app

search books (from restul API)
show a book (get book from API)
add review to the selected book (post review to API)

2)
actions on data (From the functionality we derive the actions on data)

search books (from restful API)

display books (result of searching)

select a book (from displayed books)

display the selected book (include reviews)

add review to the book

3)
possible components responsible for actions / action-data-component relation

search **books**	----> SearchForm
display **books** (book-list)	----> BookList
select a **book** from book-list	---> BookList
display the selected **book** (with its reviews)	---> BookDetail
add review to the selected **book**	----> ReviewForm

4)
shared data between components ? and common parent component ?

books shared bw. SearchForm, BookList ---> common parent == Shop
book shared bw. BookList, BookDetail, ReviewForm ---> common parent == Shop

resulting component-tree

```
Shop
    SearchForm
    BookList
    BookDetail
    ReviewForm
```

shared data in Shop component

```
books
book
```

5)
shared (display-control-) data between components ?

showBook bw. SearchForm, BookList, Shop
showReviewForm bw. ReviewForm, BookList, BookDetail, SearchForm, Shop

resulting data in Shop component

```
books
book

showBook
showReviewForm
```

6)
The Shop component looks like

```
// Shop.js

function Shop() {

  const [books, setBooks] = useState([]);
  const [book, setBook] = useState();

  const [showBook, setShowBook] = useState(false);
  const [showReviewForm, setShowReviewForm] = useState(false);

  let book_detail = <div></div>;
  let review_form = <div></div>;

  if (showBook) {
        book_detail = <BookDetail book={book} showReviewForm={showReviewForm}
        setShowReviewForm={setShowReviewForm} />;
  }

  if (showReviewForm) {
        review_form = <ReviewForm book={book} setShowReviewForm={setShowReviewForm}/> ;
  }
```

```
    return (
      <div>
          <div className="row">
            <SearchForm setBooks={setBooks} setShowBook={setShowBook}
            setShowReviewForm={setShowReviewForm}  />
          </div>
          <div className="row">
            <div className="col-sm-12 col-md-6">
              <BookList books={books} setBook={setBook} setShowBook={setShowBook}
              setShowReviewForm={setShowReviewForm}/>
            </div>
            <div className="col-sm-12 col-md-6">
              { review_form }
              { book_detail }
            </div>
          </div>
      </div>
    );

}
```

or

```
const shopInitialState = {
    books: [],
    book: null,
    showDetail: false,
    showReviewForm: false
};

function shopReducer(state, action) {
    switch(action.type){
      case 'filter':
      case 'showBook':
        return {...state, ...action.payload}
      case 'addReview':
      case 'showReviewForm':
        return {...state, showReviewForm: action.payload}
      default:
        return shopInitialState
    }
}

function Shop() {

    const [state, dispatch] = useReducer(shopReducer, shopInitialState)

    let book_detail = <div></div>;
    let review_form = <div></div>;

    if (state.showReviewForm) {
            review_form = <ReviewForm book={state.book} dispatch={dispatch} /> ;
    }
    if (state.showDetail) {
            book_detail = <BookDetail book={state.book} showReviewForm={state.showReviewForm}
            dispatch={dispatch} /> ;
    }

    return (
      <div>
        <div className="row">
          <SearchForm dispatch={dispatch} />
        </div>
        <div className="row">
          <BookList books={state.books} dispatch={dispatch} />
          ...
          { review_form }
          { book_detail }
        </div>
      </div>
    );

  }
```

Result: Implement **Version-1 of "Book Review" UI app**

https://github.com/hong1234/BookReviewReactUI_v1   or
https://github.com/hong1234/BookReviewReactUI_v1b

ReFactoring and New Implementation the UI App using the React Router

With the React Router features

a)
Whether a component is "mount" or " unmount" is controlled via the browser URL. When "mount", the state of the component is reset to "default" values.

An example, the BookDetail component (and its parent Shop component) will be "mount" if the browser URL is, for example, "/books/1" because it matches the pattern "/books/:bookId" defined in the route path.

```
<Routes>
  <Route path='/' element={<Layout />}>
    <Route path='books' element={<Shop/>}>
      <Route path=':bookId' element={<BookDetail/>} />
```

Consequence of a), the variables "showBook", "showReviewForm " in the version 1 are no longer necessary.

b)
The parent route component can forward data to the child route component via the "context" property of the <Outlet /> element and useOutletContext hook. E.g.

```
// Shop.js
function Shop() { // parent-route's component
  const [book, setBook] = useState();
  return (
    ...
    <Outlet context={[book]} />

// BookDetail.js
import { useOutletContext } from 'react-router-dom';
const BookDetail = () => { // child-route's component
  const [book] = useOutletContext()
```

c)
The dynamic route component can get data via URL parameter using useParams hook.
E.g. the BookDetail is defined in the dynamic route with path "*:bookId*", we can get value of bookId

```
// BookDetail
import { useParams } from 'react-router-dom';
const BookDetail = () => {
  const {bookId} = useParams();
```

Consequence of c), the variables "book", "setBook" in Shop component are no longer necessary.

we can alter/improve the above UI App as follows

1)
we structure the UI in 2 layers defined by 2 nested route components Shop and BookDetail

Shop layer with Route '/books'
is responsible for searching and displaying the books

```
Shop
  SearchForm
  BookList
```

BookDetail layer with Route '/books/:bookId'
is responsible for displaying the specific book and adding new reviews to the book

```
BookDetail
  ReviewForm
```

2)
the route of Shop is parent-route in relation to the route of BookDetail

```
// App.js
export default function App() {
    …
    return (
        <div>
            <Routes>
                <Route path='/' element={<Layout />}>
                    <Route index element={<Home/>} />
                    <Route path='books' element={<Shop/>}>
                        <Route path=':bookId' element={<BookDetail/>} />
                        ...
                    </Route>
                </Route>
                …
            </Routes>
        </div>
    )
}
```

3)
add <Outlet /> to JSX-code of Shop (parent)

```
// Shop.js
function Shop() {
  const [books, setBooks] = useState([]);
  ...

  return (
    <div>
      <SearchForm setBooks={setBooks}/>
      <BookList books={books} />
      ...
      <Outlet />
```

define Links to childs in parent

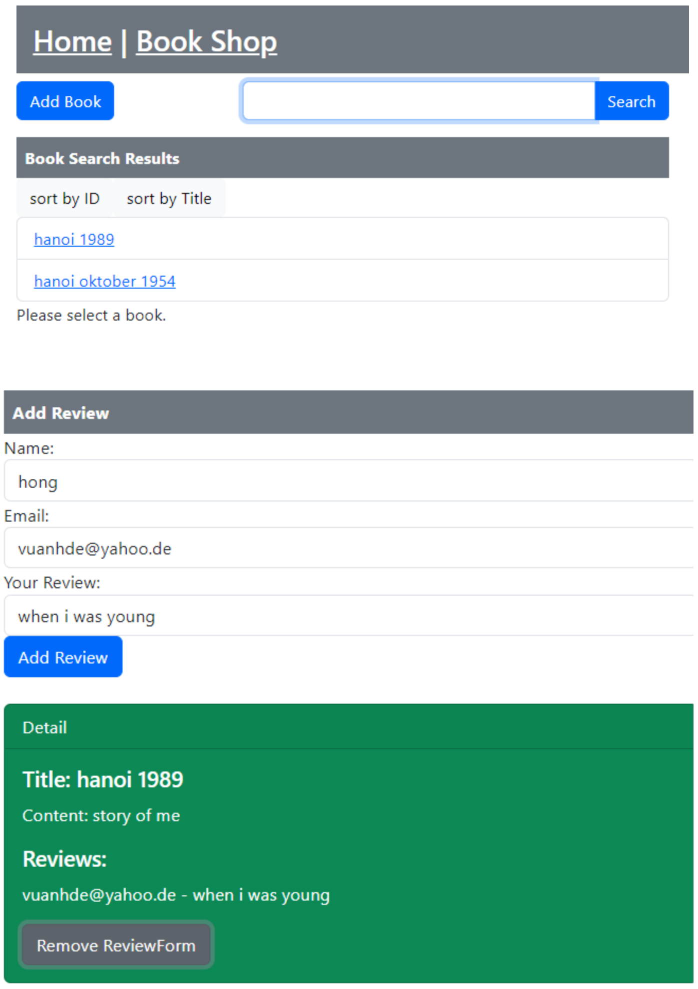
```
// BookList.js file
const linkList = sortedList.map((book) => {
  return (
    <li key={book.id} className="list-group-item">
      <Link to={`${book.id}`} >{book.title}</Link>
    </li>
  );
 });

return (
  <div >
    <ul className="list-group">
     {linkList}
    </ul>
   </div>
```

4)
get parameters or context (data) in BookDetail (child) via hooks

```
// BookDetail.js file
const BookDetail = () => {

  const {bookId} = useParams();
  ...
  const [book, setBook] = useState();  // null  default
  const [showReviewForm, setShowReviewForm] = useState(false);

  useEffect(() => {
    if (bookId !== lastId){
      getBook(bookId);
    }
   });

 return (
   <div>
     <ReviewForm book={book} .../>
     <div >
       <div className="card-header">Detail</div>
       <h5 className="card-title">Title: {book.title}</h5>
       <p className="card-text">Content: {book.content}</p>
       ...
```

=> result: **Version-2 of "Book Review" UI app**

https://github.com/hong1234/BookReviewReactUI_v2



Book Search Results

sort by ID sort by Title

hanoi 1989

hanoi oktober 1954

Please select a book.

Add Review

Name: hong

Email: vuanhde@yahoo.de

Your Review: when i was young

Detail

Title: hanoi 1989

Content: story of me

Reviews:

vuanhde@yahoo.de - when i was young