React Fundamentals

React Fundamentals

React Course

My React Course

Support

Find the Content Useful? You can always buy me a coffee

Folder Structure

- node_modules
  Contains all dependencies required by the app. Main dependencies also listed in package.json
- public
  Contains static assets including index.html (page template)
  - index.html
    - title
    - fonts
    - css
    - favicon
    - id="root" - our entire app
- src
  In simplest form it's the brain of our app. This is where we will do all of our work. src/index.js is the JavaScript entry point.
- .gitignore
  Specifies which files source control (Git) should ignore
- package.json
  Every Node.js project has a package.json and it contains info about our project, for example list of dependencies and scripts
- package-lock.json
  A snapshot of the entire dependency tree
- README
  The markdown file where you can share more info about the project for example build instructions and summary
- zoom 175%

Remove Boilerplate

- remove src folder
- create src folder
    - create index.js inside src
- toggle sidebar CMD + B
- shortcuts settings/keyboard shortcuts

First Component

```
function Greeting() {
  return <h2>My First Component</h2>;
}

// arrow function also works

const Greeting = () => {
  return <h2>My First Component</h2>;
};
```

- starts with capital letter
- must return JSX (html)
- always close tag

Typical Component

```
// imports or logic

const Greeting = () => {
  return <h2>My First Component</h2>;
};
export default Greeting;
```

Root Component (only one)

index.js

```
import React from "react";
import ReactDOM from "react-dom/client";

function Greeting() {
  return <h2>My First Component</h2>;
}

const root = ReactDOM.createRoot(document.getElementById("root"));

root.render(<Greeting />);
```

Possible Bug

If for some reason you still have this error in the terminal

```
Module not found: Error: Can't resolve 'path/index.js'
```

Just restart the server

- CTRL + C (stop the server)
- "npm start" (start the dev server)

Extensions and settings.json

- Auto Rename Tag
- Highlight Matching Tag
  - customize in settings.json
- Prettier
  - format on save
  - format on paste
  - Default Formatter (Prettier - Code formatter)

settings.json

```
  "editor.formatOnPaste": true,
  "editor.formatOnSave": true,
  "editor.defaultFormatter": "esbenp.prettier-vscode",
    "prettier.singleQuote": true,
    "prettier.semi": false,
```

- Emmet

settings.json

```
"emmet.includeLanguages": {
    "javascript": "javascriptreact"
  },
```

- ES7 Snippets
  - rafce (arrow func with export)
  - rfce (regular func with export)
  - same as the file name
  - react auto import
    - uncheck
    - React Snippets › Settings: Import React On Top

First Component in Detail

- capital letter
- must return something
- JSX syntax (return html)
  - to make our lives easier
  - calling function under the hood

index.js

```
const Greeting = () => {
  return React.createElement("h2", {}, "hello world");
};
```

```
function Greeting() {
  return (
    <div>
      <h2>hello world</h2>
    </div>
  );
}

const Greeting = () => {
  return React.createElement(
    "div",
    {},
    React.createElement("h2", {}, "hello world")
  );
};
```

JSX Rules

- return single element (one parent element)
  - semantics section/article
  - Fragment - let's us group elements without adding extra nodes

```
return <React.Fragment>...rest of the return</React.Fragment>;

// shorthand

return <>...rest of the return</>;
```

- camelCase property naming convention

```
return (
  <div tabIndex={1}>
    <button onClick={myFunction}>click me</button>
    <label htmlFor='name'>Name</label>
    <input readOnly={true} id='name' />
  </div>
)
// in html
<div tabindex="1">
    <button onclick="myFunction()">click me</button>
    <label for='name'>Name</label>
    <input readonly id='name' />
</div>
```

- className instead of class

```
return <div className="someValue">hello</div>;
```

- close every element

```
return <img />;
// or
return <input />;
```

- formatting
    - opening tag in the same line as return or ()

```
function Greeting() {
  return (
    <>
      <div className="someValue">
        <h3>hello people</h3>
        <ul>
          <li>
            <a href="#">hello world</a>
          </li>
        </ul>
      </div>
      <h2>hello world</h2>
      <input type="text" name="" id="" />
    </>
  );
}
```

Nest Components

```
function Greeting() {
  return (
    <div>
      <Person />
      <Message />
    </div>
  );
}

const Person = () => <h2>john doe</h2>;
const Message = () => {
  return <p>this is my message</p>;
};
```

React Developer Tools

- top right corner
- more tools/extensions
- open chrome web store

Book List

- setup structure

```
import React from "react";
import ReactDOM from "react-dom/client";

function BookList() {
  return (
    <section>
      <Book />
      <Book />
      <Book />
      <Book />
    </section>
  );
}

const Book = () => {
  return (
    <article>
      <Image />
      <Title />
      <Author />
    </article>
```

```
  );
};

const Image = () => <h2>image placeholder</h2>;
const Title = () => {
  return <h2>Book Title</h2>;
};
const Author = () => <h4>Author</h4>;

const root = ReactDOM.createRoot(document.getElementById("root"));

root.render(<BookList />);
```

- in search engine type - 'amazon best selling books'
  Amazon Best Sellers
- DON'T NEED TO BUY ANYTHING !!!
- NOT AN AFFILIATE LINK !!!!
- choose a book
- copy image, title and author

```
import React from "react";
import ReactDOM from "react-dom/client";

function BookList() {
  return (
    <section>
      <Book />
      <Book />
      <Book />
      <Book />
    </section>
  );
}

const Book = () => {
  return (
    <article className="book">
      <Image />
      <Title />
      <Author />
    </article>
  );
};
```

```
const Image = () => (
  <img
    src="https://images-na.ssl-images-
amazon.com/images/I/71m+Qtq+HrL._AC_UL900_SR900,600_.jpg"
    alt="Interesting Facts For Curious Minds"
  />
);
const Title = () => {
  return <h2>Interesting Facts For Curious Minds</h2>;
};
const Author = () => <h4>Jordan Moore </h4>;

const root = ReactDOM.createRoot(document.getElementById("root"));

root.render(<BookList />);
```

CSS

- create index.css in src

```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

body {
  font-family: system-ui, -apple-system, BlinkMacSystemFont, "Segoe UI",
Roboto,
    Oxygen, Ubuntu, Cantarell, "Open Sans", "Helvetica Neue", sans-serif;
  background: #f1f5f8;
  color: #222;
}
```

- import file and add classes

```
import "./index.css";

function BookList() {
  return (
    <section className="booklist">
      <Book />
      <Book />
      <Book />
```

```
      <Book />
    </section>
  );
}

const Book = () => {
  return (
    <article className="book">
      <Image />
      <Title />
      <Author />
    </article>
  );
};
```

- complete css

```
.booklist {
  width: 90vw;
  max-width: 1170px;
  margin: 5rem auto;
  display: grid;
  gap: 2rem;
}

@media screen and (min-width: 768px) {
  .booklist {
    grid-template-columns: repeat(3, 1fr);
  }
}
.book {
  background: #fff;
  border-radius: 1rem;
  padding: 2rem;
  text-align: center;
}
.book img {
  width: 100%;
  object-fit: cover;
}
.book h2 {
  margin-top: 1rem;
  font-size: 1rem;
}
```

Local Images (Public Folder)

- Optional Video !!!
- external images (hosted on different server) - just need an url
- local images (public folder) - less performant
- local images (src folder) - better solution for assets,
  since under the hood they get optimized.
- save image (Save Image As....)
- create images folder in public
- copy/paste image
- rename (optional)
- replace url in the src - './images/imageName.extension'
- './' because assets are on the same server

```
const Image = () => (
  <img src="./images/book-1.jpg" alt="Interesting Facts For Curious Minds"
/>
);
```

- whatever assets we place in public - instantly available
- domain(localhost)/asset

JSX - CSS (inline styles)

- style prop
- {} in JSX means going back to JS Land
- value is an object with key/value pairs - capitalized and with ''

```
const Author = () => (
  <h4 style={{ color: "#617d98", fontSize: "0.75rem", marginTop: "0.5rem"
}}>
    Jordan Moore
  </h4>
);
```

- css rules still apply (inline vs external css)

```
.book h4 {
  /* won't work */
  color: red;
  /* will work */
  letter-spacing: 2px;
}
```

- external libraries use inline css,
  so if you want to make some changes,
  reference the library docs and elements tab
- alternative option

```
const Author = () => {
  const inlineHeadingStyles = {
    color: "#617d98",
    fontSize: "0.75rem",
    marginTop: "0.5rem",
  };
  return <h4 style={inlineHeadingStyles}>Jordan Moore </h4>;
};
```

- FOR THE MOST PART, MULTIPLE APPROACHES AVAILABLE !!!
- AS LONG AS THE RESULT IS THE SAME, REALLY COMES DOWN TO PREFERENCE !!!!

JSX - Javascript

- refactor to single book component (personal preference)
- remove inline css

```
const Book = () => {
  return (
    <article className="book">
      <img
        src="./images/book-1.jpg"
        alt="Interesting Facts For Curious Minds"
      />
      <h2>Interesting Facts For Curious Minds</h2>
      <h4>Jordan Moore </h4>
    </article>
  );
};
```

```
.book h4 {
  color: #617d98;
  font-size: 0.75rem;
  margin-top: 0.5rem;
  letter-spacing: 2px;
}
```

- {} in JSX means going back to JS Land
- value inside must be an expression (return value),
  can't be a statement

```
const author = "Jordan Moore";
const Book = () => {
  const title = "Interesting Facts For Curious Mindssssss";
  return (
    <article className="book">
      <img
        src="./images/book-1.jpg"
        alt="Interesting Facts For Curious Minds"
      />
      <h2>{title}</h2>

      <h4>{author.toUpperCase()} </h4>
      {/* <p>{let x = 6}</p> */}
      <p>{6 + 6}</p>
    </article>
  );
};
```

- toggle line comment Edit/Toggle Line Comment

Props - Initial Setup

- refactor/clean up

```
const author = "Jordan Moore";
const title = "Interesting Facts For Curious Minds";
const img = "./images/book-1.jpg";

function BookList() {
  return (
    <section className="booklist">
      <Book />
      <Book />
    </section>
  );
}
const Book = () => {
  return (
    <article className="book">
      <img src={img} alt={title} />
      <h2>{title}</h2>
      <h4>{author} </h4>
    </article>
  );
};
```

```
// parameters
const someFunc = (param1, param2) => {
  console.log(param1, param2);
};
// arguments
someFunc("job", "developer");
```

```
const Book = (props) => {
  console.log(props);
  return (
    <article className="book">
      <img src={img} alt={title} />
      <h2>{title}</h2>
      <h4>{author} </h4>
      {console.log(props)}
    </article>
  );
};
```

- props object, convention to call props, 'shakeAndBake' is an excellent alternative
- pass as key/value pairs
- if the prop exists it will return value, otherwise no value

```
function BookList() {
  return (
    <section className="booklist">
      <Book job="developer" />
      <Book title="random title" number={22} />
    </section>
  );
}
const Book = (props) => {
  console.log(props);
  return (
    <article className="book">
      <img src={img} alt={title} />
      <h2>{title}</h2>
      <h4>{author} </h4>
      <p>{props.job}</p>
      <p>{props.title}</p>
      <p>{props.number}</p>
    </article>
  );
};
```

```
function BookList() {
  return (
    <section className="booklist">
      <Book author={author} title={title} img={img} />
      <Book title={title} img={img} />
    </section>
  );
}
const Book = (props) => {
  console.log(props);
  return (
    <article className="book">
      <img src={props.img} alt={props.title} />
      <h2>{props.title}</h2>
      <h4>{props.author} </h4>
    </article>
  );
};
```

Props - Somewhat Dynamic Setup

- setup an object
- refactor vars to properties
- copy/paste and rename
- get values for second book
- setup props

```
const firstBook = {
  author: "Jordan Moore",
  title: "Interesting Facts For Curious Minds",
  img: "./images/book-1.jpg",
};
const secondBook = {
  author: "James Clear",
  title: "Atomic Habits",
  img: "https://images-na.ssl-images-
amazon.com/images/I/81wgcld4wxL._AC_UL900_SR900,600_.jpg",
};

function BookList() {
  return (
    <section className="booklist">
      <Book
```

```
        author={firstBook.author}
        title={firstBook.title}
        img={firstBook.img}
      />
      <Book
        author={secondBook.author}
        title={secondBook.title}
        img={secondBook.img}
      />
    </section>
  );
}
const Book = (props) => {
  console.log(props);
  return (
    <article className="book">
      <img src={props.img} alt={props.title} />
      <h2>{props.title}</h2>
      <h4>{props.author} </h4>
    </article>
  );
};
```

Access Props - Multiple Approaches

- there is no right or wrong - again preference !!!
- Destructuring (object)
  JS Nuggets - Destructuring (object)
- destructuring in Vanilla JS
- saves time/typing
- pull out the properties
- don't need to reference object anymore

```
const someObject = {
  name: "john",
  job: "developer",
  location: "florida",
};

console.log(someObject.name);
const { name, job } = someObject;
console.log(job);
```

- no need for all the props.propName

- destructure inside component

```
const Book = (props) => {
  const { img, title, author } = props;
  return (
    <article className="book">
      <img src={img} alt={title} />
      <h2>{title}</h2>
      <h4>{author} </h4>
    </article>
  );
};
```

- destructure in function parameters (in our case props)
- if you have console.log(props) - it won't be defined

```
const Book = ({ img, title, author }) => {
  return (
    <article className="book">
      <img src={img} alt={title} />
      <h2>{title}</h2>
      <h4>{author} </h4>
    </article>
  );
};
```

Children Prop

- everything we render between component tags
- during the course we will mostly use it Context API
- special prop, has to be "children"
- can place anywhere in JSX

```
function BookList() {
  return (
    <section className="booklist">
      <Book
        author={firstBook.author}
        title={firstBook.title}
        img={firstBook.img}
      >
        <p>
          Lorem ipsum dolor, sit amet consectetur adipisicing elit. Itaque
          repudiandae inventore eos qui animi sed iusto alias eius ea
sapiente.
```

```
        </p>
        <button>click me</button>
      </Book>
      <Book
        author={secondBook.author}
        title={secondBook.title}
        img={secondBook.img}
      />
    </section>
  );
}

const Book = ({ img, title, author, children }) => {
  // rest of the logic
};
const Book = (props) => {
  const { img, title, author, children } = props;
  console.log(props);
  return (
    <article className="book">
      <img src={img} alt={title} />
      <h2>{title}</h2>
      <h4>{author} </h4>
      {children}
    </article>
  );
};
```

- optional

```
@media screen and (min-width: 768px) {
  .booklist {
    grid-template-columns: repeat(3, 1fr);
    align-items: start;
  }
}
.book p {
  margin: 1rem 0 0.5rem;
}
```

Simple List

- Javascript Nuggets - Map
- refactor

```
const books = [
  {
    author: "Jordan Moore",
    title: "Interesting Facts For Curious Minds",
    img: "./images/book-1.jpg",
  },
  {
    author: "James Clear",
    title: "Atomic Habits",
    img: "https://images-na.ssl-images-
amazon.com/images/I/81wgcld4wxL._AC_UL900_SR900,600_.jpg",
  },
];

function BookList() {
  return <section className="booklist"></section>;
}

const Book = (props) => {
  const { img, title, author } = props;

  return (
    <article className="book">
      <img src={img} alt={title} />
      <h2>{title}</h2>
      <h4>{author} </h4>
    </article>
  );
};
```

- can't render objects in React

```
function BookList() {
  return <section className="booklist">{books}</section>;
}
```

- map - creates a new array from calling a function for every array element.

```
const names = ["john", "peter", "susan"];
const newNames = names.map((name) => {
  console.log(name);
  return <h1>{name}</h1>;
});

function BookList() {
```

```
  return <section className="booklist">{newNames}</section>;
}
```

Proper List

- remove names and newNames

```
function BookList() {
  return (
    <section className="booklist">
      {books.map((book) => {
        console.log(book);

        // return 'hello';
        return (
          <div>
            <h2>{book.title}</h2>
          </div>
        );
      })}
    </section>
  );
}
```

- render component
- pass properties one by one

```
function BookList() {
  return (
    <section className="booklist">
      {books.map((book) => {
        console.log(book);
        const { img, title, author } = book;
        return <Book img={img} title={title} author={author} />;
      })}
    </section>
  );
}
```

Key Prop

- typically it's going to be id

```
const books = [
  {
    author: "Jordan Moore",
```

```
    title: "Interesting Facts For Curious Minds",
    img: "./images/book-1.jpg",
    id: 1,
  },
  {
    author: "James Clear",
    title: "Atomic Habits",
    img: "https://images-na.ssl-images-
amazon.com/images/I/81wgcld4wxL._AC_UL900_SR900,600_.jpg",
    id: 2,
  },
];

function BookList() {
  return (
    <section className="booklist">
      {books.map((book) => {
        console.log(book);
        const { img, title, author, id } = book;
        return <Book book={book} key={id} />;
      })}
    </section>
  );
}
```

- you will see index,but it's not advised if the list is changing

```
function BookList() {
  return (
    <section className="booklist">
      {books.map((book, index) => {
        console.log(book);
        const { img, title, author, id } = book;
        return <Book book={book} key={index} />;
      })}
    </section>
  );
}
```

Pass The Entire Object

- render component
- pass entire object

- Destructuring (object)
  JS Nuggets - Destructuring (object)

```
function BookList() {
  return (
    <section className="booklist">
      {books.map((book) => {
        console.log(book);
        const { img, title, author } = book;
        return <Book book={book} />;
      })}
    </section>
  );
}

const Book = (props) => {
  const { img, title, author } = props.book;

  return (
    <article className="book">
      <img src={img} alt={title} />
      <h2>{title}</h2>
      <h4>{author} </h4>
    </article>
  );
};
```

- alternative

```
const Book = ({ book: { img, title, author } }) => {
  return (
    <article className="book">
      <img src={img} alt={title} />
      <h2>{title}</h2>
      <h4>{author} </h4>
    </article>
  );
};
```

My Personal Preference

- utilize spread operator (...) - copy values
- Spread Operator
- JS Nuggets - Spread Operator

```
const friends = ["john", "peter", "anna"];
const newFriends = [...friends, "susan"];
console.log(friends);
console.log(newFriends);
const someObject = {
  name: "john",
  job: "developer",
};
// COPY NOT A REFERENCE !!!!
const newObject = { ...someObject, location: "florida" };
console.log(someObject);
console.log(newObject);
```

```
function BookList() {
  return (
    <section className="booklist">
      {books.map((book) => {
        return <Book {...book} key={book.id} />;
      })}
    </section>
  );
}

const Book = (props) => {
  const { img, title, author } = props;
  return (
    <article className="book">
      <img src={img} alt={title} />
      <h2>{title}</h2>
      <h4>{author} </h4>
    </article>
  );
};
const Book = ({ img, title, author }) => {
  // rest of the code
};
```

Events - Fundamentals

- Vanilla JS

```
const btn = document.getElementById("btn");

btn.addEventListener("click", function (e) {
  // access event object
```

```
  // do something when event fires
});
```

- similar approach
- element, event, function
- again camelCase

```
const EventExamples = () => {
  const handleButtonClick = () => {
    alert("handle button click");
  };
  return (
    <section>
      <button onClick={handleButtonClick}>click me</button>
    </section>
  );
};
```

- [React Events]
- no need to memorize them(idea is the same)
- most common
  - onClick (click events)
  - onSubmit (submit form)
  - onChange (input change)

```
function BookList() {
  return (
    <section className="booklist">
      <EventExamples />
      {books.map((book) => {
        return <Book {...book} key={book.id} />;
      })}
    </section>
  );
}

const EventExamples = () => {
  const handleFormInput = () => {
    console.log("handle form input");
  };
  const handleButtonClick = () => {
    alert("handle button click");
  };
```

```
  return (
    <section>
      <form>
        <h2>Typical Form</h2>
        <input
          type="text"
          name="example"
          onChange={handleFormInput}
          style={{ margin: "1rem 0" }}
        />
      </form>
      <button onClick={handleButtonClick}>click me</button>
    </section>
  );
};
```

Event Object and Form Submission

```
const EventExamples = () => {
  const handleFormInput = (e) => {
    console.log(e);
    // e.target - element
    console.log(`Input Name : ${e.target.name}`);
    console.log(`Input Value : ${e.target.value}`);
    // console.log('handle form input');
  };
  const handleButtonClick = () => {
    alert("handle button click");
  };
  const handleFormSubmission = (e) => {
    e.preventDefault();
    console.log("form submitted");
  };
  return (
    <section>
      {/* add onSubmit Event Handler */}
      <form onSubmit={handleFormSubmission}>
        <h2>Typical Form</h2>
        <input
          type="text"
          name="example"
          onChange={handleFormInput}
          style={{ margin: "1rem 0" }}
        />
```

```
        {/* add button with type='submit' */}
        <button type="submit">submit form</button>
      </form>
      <button onClick={handleButtonClick}>click me</button>
    </section>
  );
};
```

- alternative approach

```
<button type="submit" onClick={handleFormSubmission}>
  submit form
</button>
```

Mind Grenade

- alternative approach
- pass anonymous function (in this case arrow function)
- one liner - less code

```
const EventExamples = () => {
  return (
    <section>
      <button onClick={() => console.log("hello there")}>click me</button>
    </section>
  );
};
```

- also can access event object

```
const EventExamples = () => {
  return (
    <section>
      <form>
        <h2>Typical Form</h2>
        <input
          type="text"
          name="example"
          onChange={(e) => console.log(e.target.value)}
          style={{ margin: "1rem 0" }}
        />
      </form>
      <button onClick={() => console.log("you clicked me")}>click
me</button>
    </section>
```

```
  );
};
```

Mind Grenade #2

- remove EventsExamples
- components are independent by default

```
function BookList() {
  return (
    <section className="booklist">
      {books.map((book) => {
        return <Book {...book} key={book.id} />;
      })}
    </section>
  );
}

const Book = (props) => {
  const { img, title, author } = props;
  const displayTitle = () => {
    console.log(title);
  };

  return (
    <article className="book">
      <img src={img} alt={title} />
      <h2>{title}</h2>
      <button onClick={displayTitle}>display title</button>
      <h4>{author} </h4>
    </article>
  );
};
```

- remove button

Prop Drilling

- react data flow - can only pass props down
- alternatives Context API, redux, other state libraries

```
function BookList() {
  const someValue = "shakeAndBake";
  const displayValue = () => {
    console.log(someValue);
  };
```

```
  return (
    <section className="booklist">
      {books.map((book) => {
        return <Book {...book} key={book.id} displayValue={displayValue} />;
      })}
    </section>
  );
}

const Book = (props) => {
  const { img, title, author, displayValue } = props;

  return (
    <article className="book">
      <img src={img} alt={title} />
      <h2>{title}</h2>
      <button onClick={displayValue}>click me</button>
      <h4>{author} </h4>
    </article>
  );
};
```

More Complex Example

- initial setup
- create getBook function in booklist
- accepts id as an argument and finds the book
- Javascript Nuggets - Filter and Find
- pass the function down to Book Component and invoke on the button click
- in the Book Component destructure id and function
- invoke the function when user clicks the button, pass the id
- the goal : you should see the same book in the console

```
const BookList = () => {
  const getBook = (id) => {
    const book = books.find((book) => book.id === id);
    console.log(book);
  };

  return (
    <section className="booklist">
      {books.map((book) => {
        return <Book {...book} key={book.id} getBook={getBook} />;
```

```
      })}
    </section>
  );
};

const Book = (props) => {
  const { img, title, author, getBook, id } = props;
  // console.log(props);

  return (
    <article className="book">
      <img src={img} alt={title} />
      <h2>{title}</h2>
      {/* this is not going to work */}
      <button onClick={getBook(id)}>display title</button>
      <h4>{author}</h4>
    </article>
  );
};
```

- two fixes
- first option - setup wrapper

```
const Book = (props) => {
  const { img, title, author, getBook, id } = props;
  // console.log(props);
  const getSingleBook = () => {
    getBook(id);
  };
  return (
    <article className="book">
      <img src={img} alt={title} />
      <h2>{title}</h2>
      <button onClick={getSingleBook}>display title</button>
      <h4>{author}</h4>
    </article>
  );
};
```

- two fixes
- second option - wrap in the anonymous arrow function

```
const Book = (props) => {
  const { img, title, author, getBook, id } = props;
```

```
  // console.log(props);
  const getSingleBook = () => {
    getBook(id);
  };
  return (
    <article className="book">
      <img src={img} alt={title} />
      <h2>{title}</h2>

      <button onClick={() => getBook(id)}>display title</button>
      <h4>{author}</h4>
    </article>
  );
};
```

Import and Export Statements

- remove all getBook code

```
function BookList() {
  return (
    <section className="booklist">
      {books.map((book) => {
        return <Book {...book} key={book.id} />;
      })}
    </section>
  );
}

const Book = (props) => {
  const { img, title, author } = props;

  return (
    <article className="book">
      <img src={img} alt={title} />
      <h2>{title}</h2>

      <h4>{author} </h4>
    </article>
  );
};
```

- setup two files in src books.js and Book.js
- cut books array from index.js

- add to books.js

books.js

```
const books = [
  {
    author: "Jordan Moore",
    title: "Interesting Facts For Curious Minds",
    img: "./images/book-1.jpg",
    id: 1,
  },
  {
    author: "James Clear",
    title: "Atomic Habits",
    img: "https://images-na.ssl-images-
amazon.com/images/I/81wgcld4wxL._AC_UL900_SR900,600_.jpg",
    id: 2,
  },
];
```

- two flavors named and default exports
  - with named exports names MUST match
  - with default exports,can rename but only one per file
- named export

```
export const books = [
  {
    author: "Jordan Moore",
    title: "Interesting Facts For Curious Minds",
    img: "./images/book-1.jpg",
    id: 1,
  },
  {
    author: "James Clear",
    title: "Atomic Habits",
    img: "https://images-na.ssl-images-
amazon.com/images/I/81wgcld4wxL._AC_UL900_SR900,600_.jpg",
    id: 2,
  },
];
```

index.js

```
import { books } from "./books";
```

- default export

```
const Book = (props) => {
  const { img, title, author } = props;

  return (
    <article className="book">
      <img src={img} alt={title} />
      <h2>{title}</h2>

      <h4>{author} </h4>
    </article>
  );
};

export default Book;
```

index.js

```
import Book from "./Book";
```

Local Images (src folder)

- better performance because optimized
- add one more book to array
- download all three images (rename)
- setup images folder in the src
- import all three images in the books.js
- set image property equal to import
- and yes each image requires new import

```
import img1 from "./images/book-1.jpg";
import img2 from "./images/book-2.jpg";
import img3 from "./images/book-3.jpg";

export const books = [
  {
    author: "Jordan Moore",
    title: "Interesting Facts For Curious Minds",
    img: img1,
    id: 1,
  },
  {
    author: "James Clear",
```

```
    title: "Atomic Habits",
    img: img2,
    id: 2,
  },
  {
    author: "Stephen King",
    title: "Fairy Tale",
    img: img3,
    id: 3,
  },
];
```

Challenges

- setup numbers
- don't worry about css
- hint - index (second parameter in map)

index.js

```
const BookList = () => {
  return (
    <section className="booklist">
      {books.map((book, index) => {
        return <Book {...book} key={book.id} number={index} />;
      })}
    </section>
  );
};

const Book = (props) => {
  const { img, title, author, number } = props;

  return (
    <article className="book">
      <img src={img} alt={title} />
      <h2>{title}</h2>

      <h4>{author}</h4>
      <span className="number">{`# ${number + 1}`}</span>
    </article>
  );
};
```

index.css

```css
.book {
  background: #fff;
  border-radius: 1rem;
  padding: 2rem;
  text-align: center;
  /* set relative */
  position: relative;
}

.number {
  position: absolute;
  top: 0;
  left: 0;
  font-size: 1rem;
  padding: 0.75rem;
  border-top-left-radius: 1rem;
  border-bottom-right-radius: 1rem;
  background: #c35600;
  color: #fff;
}
```

Add Title

- add a title to our app (css optional)
- change page title

index.js

```jsx
function BookList() {
  return (
    <>
      <h1>amazon best sellers</h1>
      <section className="booklist">
        {books.map((book) => {
          return <Book {...book} key={book.id} />;
        })}
      </section>
    </>
  );
}
```

index.css

```css
h1 {
  text-align: center;
```

```
  margin-top: 4rem;
  text-transform: capitalize;
}
```

public/index.html

```
<title>Best Sellers</title>
```

Build Production Application

- stop the dev server "ctrl + c"
- run "npm run build"
- build folder gets created

Netlify

- sign up
- add new site/deploy manually
- choose build folder
- rename site - site settings/change site name

Create-React-App Boilerplate (src)

- index.js

```
import React from "react";
import ReactDOM from "react-dom/client";

// styles (typically global)
import "./index.css";

// convention to name it App and setup in a separate file
import App from "./App";
// import report web vitals
import reportWebVitals from "./reportWebVitals";

// StrictMode

// StrictMode is a tool for highlighting potential problems in an
application.Activates additional checks and warnings for its
descendants.Runs only in Development, does not impact the production build.
RENDERS TWICE !!! Possible to remove.

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <React.StrictMode>
```

```
    <App />
  </React.StrictMode>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

- remove in src
  - setupTests.js
  - reportWebVitals.js
  - App.test.js
- be careful with multiple css files

App.js

```
function App() {
  return <h1>backroads app</h1>;
}

export default App;
```

- remove
  - remove logo.svg
  - App.css

Vite Docs

(Vite)[https://vitejs.dev/]

Vite Install

```
npm create vite@latest app-name -- --template react
npm install
npm run dev
```

- http://localhost:5173/

Vite Setup

- need to use .jsx extension
- index.html in the source instead of public
- assets still in public
- instead of index.js, need to use main.jsx

- to spin up dev server - "npm run dev"
- rest the same - imports/exports, deployment, assets, etc...