React JS essentials

Vinod Kumar Kayartaya

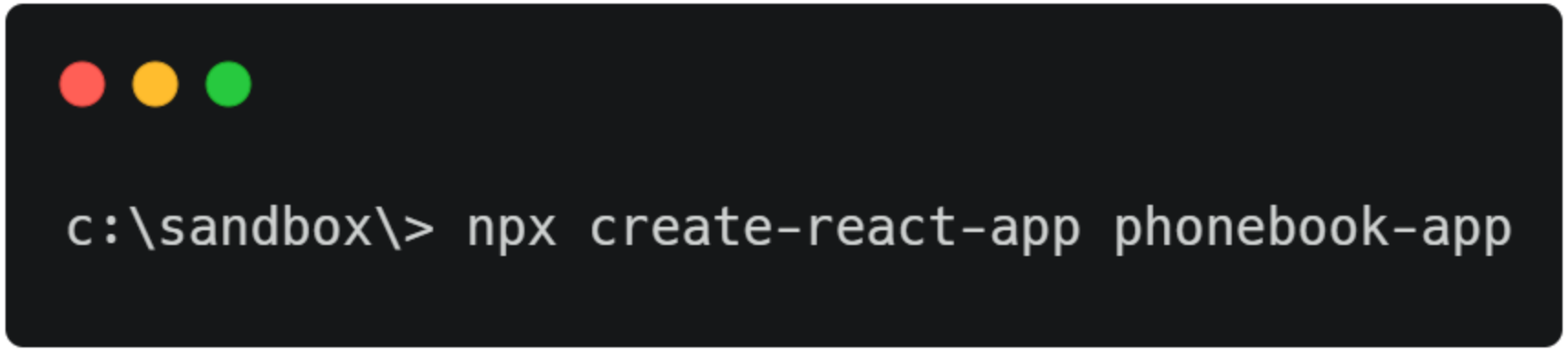
https://vinod.co

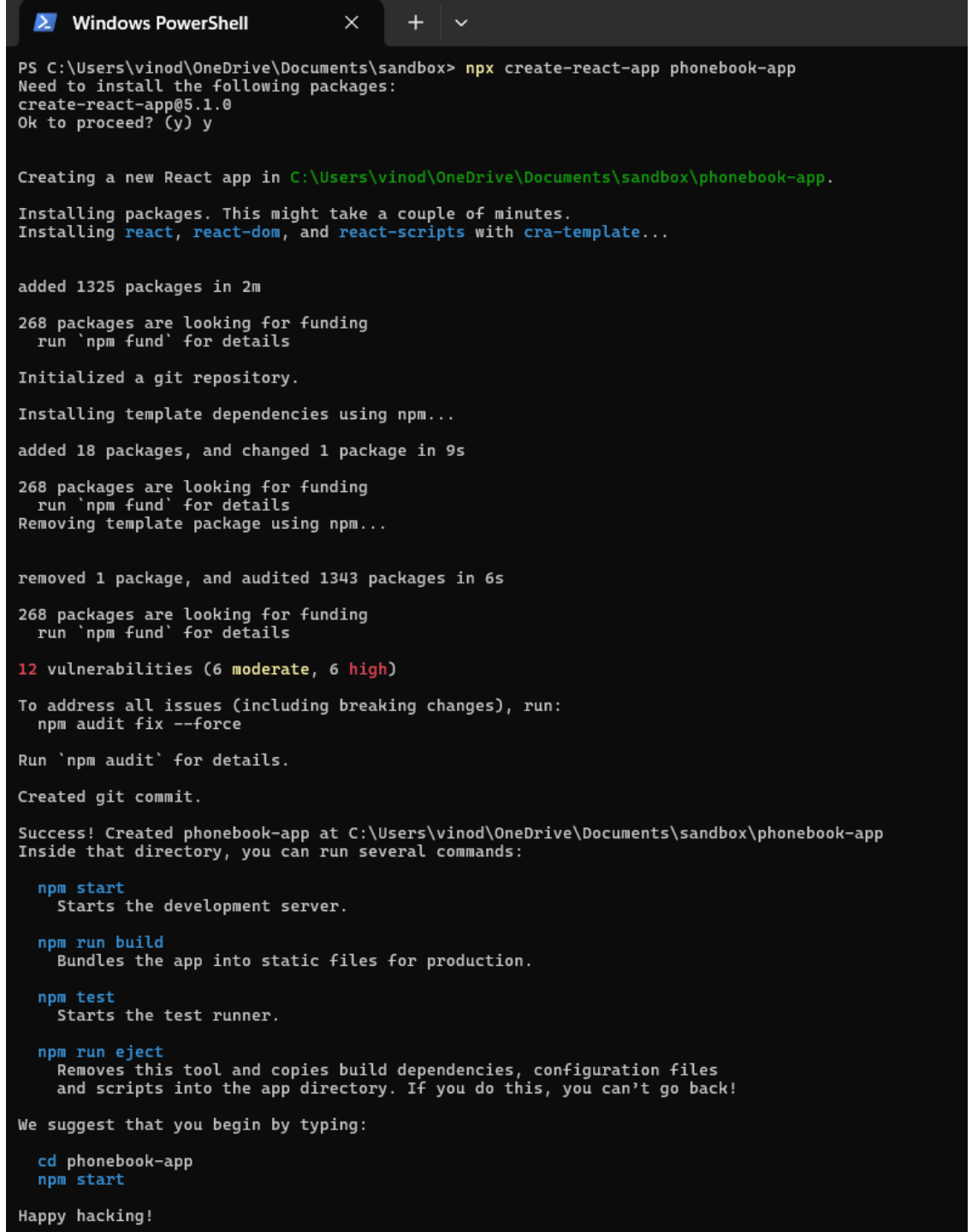
vinod@vinod.co

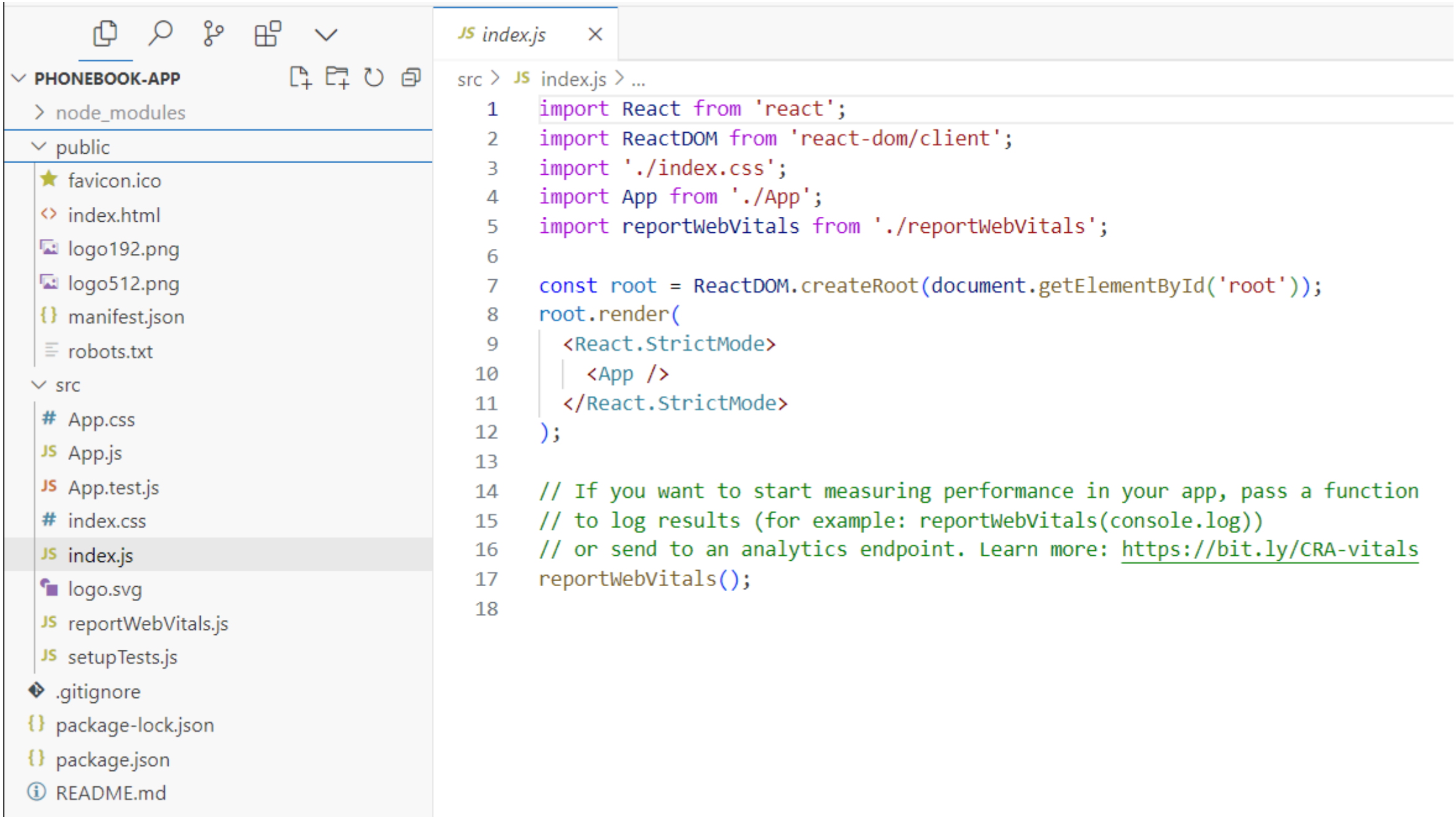
What is React JS?

- JavaScript library for building user interfaces, developed and maintained by Facebook (Meta)
- Component-based architecture that allows developers to create reusable UI elements
- Uses a virtual DOM to optimize rendering performance and improve user experience
- Follows a declarative programming paradigm, making code more predictable and easier to debug

Creating new app




Windows PowerShell
PS C:\Users\vinod\OneDrive\Documents\sandbox> npx create-react-app phonebook-app
Need to install the following packages:
create-react-app@5.1.0
Ok to proceed? (y) y
Creating a new React app in C:\Users\vinod\OneDrive\Documents\sandbox\phonebook-app.
Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...
added 1325 packages in 2m
268 packages are looking for funding
run `npm fund` for details
Initialized a git repository.
Installing template dependencies using npm...
added 18 packages, and changed 1 package in 9s
268 packages are looking for funding
run `npm fund` for details
Removing template package using npm...
removed 1 package, and audited 1343 packages in 6s
268 packages are looking for funding
run `npm fund` for details
12 vulnerabilities (6 moderate, 6 high)
To address all issues (including breaking changes), run:
npm audit fix --force
Run `npm audit` for details.
Created git commit.
Success! Created phonebook-app at C:\Users\vinod\OneDrive\Documents\sandbox\phonebook-app
Inside that directory, you can run several commands:
npm start
Starts the development server.
npm run build
Bundles the app into static files for production.
npm test
Starts the test runner.
npm run eject
Removes this tool and copies build dependencies, configuration files
and scripts into the app directory. If you do this, you can't go back!
We suggest that you begin by typing:
cd phonebook-app
npm start
Happy hacking!


PHONEBOOK-APP
node_modules
public
favicon.ico
index.html
logo192.png
logo512.png
manifest.json
robots.txt
src
App.css
App.js
App.test.js
index.css
index.js
logo.svg
reportWebVitals.js
setupTests.js
.gitignore
package-lock.json
package.json
README.md
index.js
src > index.js > ...
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
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();

What is a component?

- Basic building block of React applications, encapsulating functionality and UI elements
- Self-contained, reusable pieces of code that can maintain their own state
- Can be functional (using hooks) or class-based, both rendering UI elements
- Can accept inputs (props) and return React elements describing what should appear on screen

A basic component

```
const HelloWorld = () => {
  return <h1>Hello, World!</h1>;
};

export default HelloWorld;
```

How to use it?

```
import React from "react";
import HelloWorld from "./HelloWorld"; // Import the component

const App = () => {
  return (
    <div>
      <HelloWorld />
    </div>
  );
};

export default App;
```

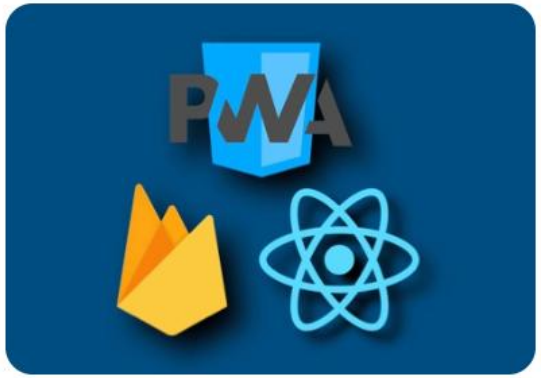
Component architecture

- Hierarchical tree structure with parent-child relationships between components
- Unidirectional data flow (top-down) making applications more predictable and easier to debug
- Component composition allows complex UIs to be built from simple, isolated pieces
- Props and state management facilitate communication between components

Watch my FULL courses on YouTube

Spring Framework MasterClass - from basics to advanced

Developing a RESTful web service using spring boot



Learn React by building a progressive

Python 3 Programming MasterClass -

Consuming REST services using the CPR

Discover how to seamlessly integrate REST services into your C++ projects using the CPR library. With CPR's intuitive API, handling HTTP requests becomes straightforward, whether you're fetching data, posting updates, or managing authentication. Learn how to harness CPR's features, from custom headers and authentication methods to handling redirects and timeouts, empowering your applications with efficient communication with RESTful APIs. Dive into the world of modern C++ development and elevate your projects by consuming REST services effortlessly with CPR.

Developing RESTful services using Crow CPP

Dive into the world of creating robust and efficient RESTful APIs with Crow CPP, a lightweight C++ microframework for web development. We explore designing RESTful endpoints, handling requests and responses, and implementing middleware. This comprehensive guide equips you with the knowledge and tools needed to build scalable RESTful services using Crow CPP.

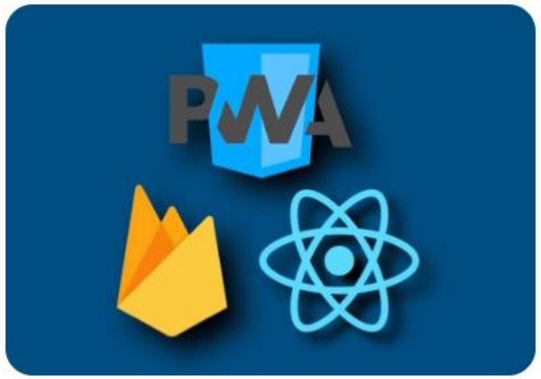
ReST

REST, or Representational State Transfer, is an architectural style for designing networked applications. It was introduced by Roy Fielding in his doctoral dissertation in 2000. RESTful systems

Watch my FULL courses on YouTube

Spring Framework MasterClass - from basics to advanced

Developing a RESTful web service using spring boot



Learn React by building a progressive

Python 3 Programming MasterClass -

Consuming REST services using the CPR

Discover how to seamlessly integrate REST services into your C++ projects using the CPR library. With CPR's intuitive API, handling HTTP requests becomes straightforward, whether you're fetching data, posting updates, or managing authentication. Learn how to harness CPR's features, from custom headers and authentication methods to handling redirects and timeouts, empowering your applications with efficient communication with RESTful APIs. Dive into the world of modern C++ development and elevate your projects by consuming REST services effortlessly with CPR.

Developing RESTful services using Crow CPP

Dive into the world of creating robust and efficient RESTful APIs with Crow CPP, a lightweight C++ microframework for web development. We explore designing RESTful endpoints, handling requests and responses, and implementing middleware. This comprehensive guide equips you with the knowledge and tools needed to build scalable RESTful services using Crow CPP.

ReST

REST, or Representational State Transfer, is an architectural style for designing networked applications. It was introduced by Roy Fielding in his doctoral dissertation in 2000. RESTful systems

Component hierarchy

```
App
├── Header
├── MainContent
│   ├── CoursesSection
│   │   ├── CourseCard (reusable)
│   ├── ArticlesSection
│       ├── ArticleCard (reusable)
├── Footer (if applicable)
```

JSX

- Syntax extension that allows HTML-like code within JavaScript
- Transforms HTML tags into React elements through a compilation step
- Expressions can be embedded using curly braces {expression}
- Makes React components more readable and intuitive to write

Example JSX

```jsx
const CourseCard = ({ title, image }) => (
  <div className="course-card">
    <img src={image} alt={title} />
    <h3>{title}</h3>
  </div>
);
```

Prop

- Read-only data passed from parent to child components
- Enables component reusability by allowing customization of child components
- Cannot be modified by the receiving component (one-way data flow)
- Can contain any JavaScript value including functions, objects, and other React elements

State

- Internal data storage that determines a component's behavior and rendering
- When state changes, the component re-renders to reflect updated information
- Managed with useState hook in functional components or this.state in class components
- Should be treated as immutable, with updates handled through setState or state updater functions

Virtual DOM

- Lightweight copy of the actual DOM that React maintains in memory
- Minimizes direct DOM manipulation by comparing virtual DOM with real DOM
- Performs batch updates to reduce costly browser repainting operations
- Significantly improves performance by only updating what has changed