React

npm download package

first file is processed -package.json

set port =8000 to set custom port

^verison set an base version

index.js is like main function

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

index has root element it create

```
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

app is user defined tag

react js we can define our own tags

html is limited in tags

import is used to

two type of compnents

functional , functional tag —

class tag

Understanding React Props

Definition:

- Props (short for properties) are a fundamental concept in React that allows data to be passed from one component to another.

- They are similar to attributes for HTML tags.

Passing Props:

- To pass props to a component, we declare them as attributes for the component just like HTML tags.
- Syntax: `<ComponentName propName="value" />`

Default Props:

- You can also define default values for props in a component.
- Syntax:

```
javascriptCopy code
ComponentName.defaultProps = {
  propName: "defaultValue"
}
```

Accessing Props:

- Inside the receiving component, you can access props using `this.props.propName`.

Pass by Value:

- Props in React are passed by value.
- This means that changes made to props within a child component won't affect the original data in the parent component.
- This behavior is similar to how primitive data types are passed in JavaScript.

Example:

Suppose we have a `Person` component that receives a `name` prop:

```
javascriptCopy code
// Passing props
<Person name="John" />

// Accessing props
class Person extends React.Component {
  render() {
    return <h1>{this.props.name}</h1>;
  }
}
```

Default Props Example:

```
javascriptCopy code
class Person extends React.Component {
  // Setting default value for name prop
  static defaultProps = {
    name: "Guest"
  };

  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}

// Using the component
<Person /> // Renders: Hello, Guest
<Person name="John" /> // Renders: Hello, John
```

In this example, if the `name` prop is not provided, it defaults to "Guest."

States vs. Props in React

React uses both `state` and `props` to manage and transfer data within a component-based application. Understanding the differences between them is crucial for developing effective and maintainable React applications.

Props (Properties):

1. **Immutable**: Props are read-only. They are passed from parent to child components and cannot be modified by the child component.
2. **External Data**: Props are used to pass data from a parent component to a child component. They are external inputs to the component.
3. **Usage**: Props are suitable for transferring data that should not be modified within the child component, such as configuration options, data fetched from APIs, or data passed from a higher-level component.
4. **Syntax**: Props are declared in the JSX of the parent component and accessed using `this.props` in the child component.
5. **Example**:

```
javascriptCopy code
<ChildComponent data={someData} />
```

```
// Access in ChildComponent: this.props.data
```

State:

1. **Mutable**: State is mutable and belongs to the component that defines it. It can be changed using the `setState` method.
2. **Local Data**: State is used to manage internal data within a component. It represents data that can change over time and affect the component's rendering.
3. **Usage**: State is ideal for managing component-specific data, such as form input values, UI state (e.g., whether a dropdown is open or closed), or any data that needs to be updated and trigger re-rendering.
4. **Syntax**: State is declared using the `constructor` method within a class component and accessed using `this.state`.
5. **Example**:

```
javascriptCopy code
constructor(props) {
  super(props);
  this.state = {
    count: 0
  };
}

// Access and update state
this.setState({ count: this.state.count + 1 });
```

Key Differences:

1. **Mutability**: Props are immutable (read-only), while state is mutable (can be changed).
2. **Ownership**: Props are owned by the parent component and passed down to child components. State is owned by the component that defines it and cannot be accessed or modified directly by other components.
3. **Scope**: Props have a broader scope, as they can be accessed by child components. State is local and specific to the component in which it is defined.
4. **Changes and Re-renders**: Changes to props do not cause a component to re-render, while changes to state trigger re-renders.

There's no need for `this` when working with functional components because they don't have a `this` context like class components do. Props are simply passed as an argument to the function, making it more straightforward to work with.

props in React and attributes/values in HTML share a similarity in how they are declared as key-value pairs, props in React are specifically designed for passing data and configuration between components within a React application, and they can hold dynamic values that can change over time. In contrast, attributes and values in HTML are primarily used for configuring and styling elements in a static manner.

use to mention defaults in tags and appilcation

learn the react routing