Reactjs

week 3

Margit Tennosaar

Last session

- Components
- Introduction to Props and State
- Handling events
- Basic Hooks usage (useState)
- List and keys

This session

- Immutable JS
- Forms and controlled components
- Managing lists and keys
- Conditional rendering techniques

Immutable JS

Why mutating in react is not recommended?

If you change state directly, then React can’t figure out that the state of a component has changed, and component will not be updated.

Immutability, allows React to compare the old state if an object with its new state and re-render the component based on that difference.

https://css-tricks.com/understanding-immutability-in-javascript/

Arrays

JavaScript arrays are used to store multiple values in a single variable

JavaScript offers several ways to add, remove, and replace items in an array – but some of these ways mutate the array, and others are immutable - they produce a new array.

Some of the most common array methods

Add

- array.push()
- array.unshift()
- ★ array.concat()
- ★ Spread operator

Replace

- array.splice()
- ★ array.map()

Remove

- array.pop()
- array.shift()
- array.splice()
- ★ array.filter()
- ★ array.slice()

★ - immutable methods

https://lorenstewart.me/2017/01/22/javascript-array-methods-mutating-vs-non-mutating/

Unidirectional Data flow

props contains information set by the parent component (although defaults can be set) and should not be changed.

state contains “private” information for the component to initialise, change, and use on it’s own.

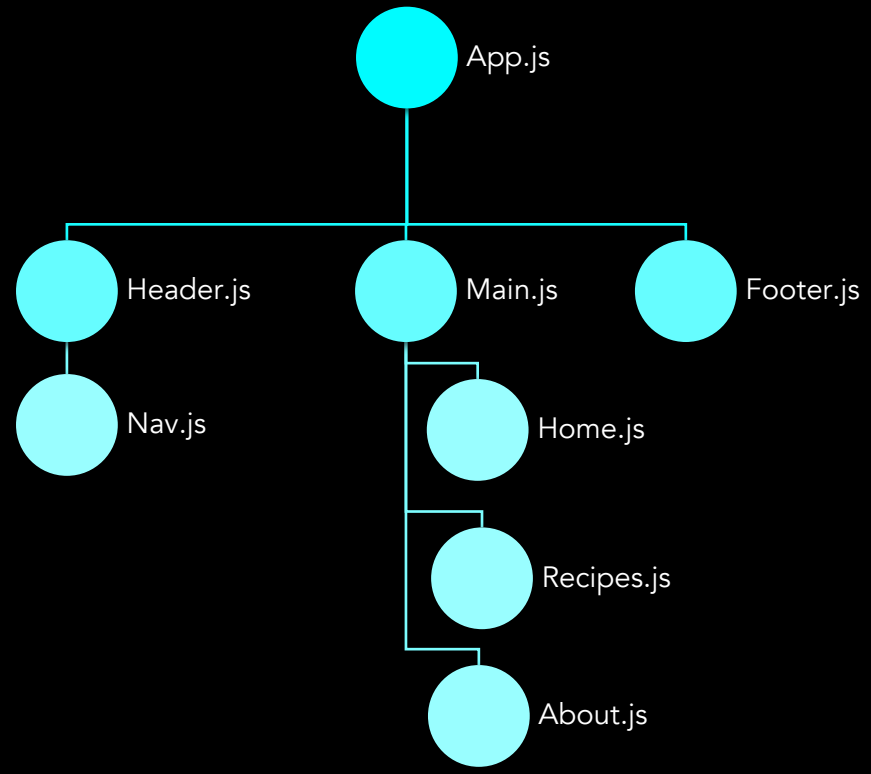
setState() schedules an update to a component’s state object. When state changes, the component responds by re-rendering.

Unidirectional Data Flow

A state is always owned by one Component.

Changing the state on a Component will not affect its parents, or siblings or any other Component in the application: just its children

This is the reason why state is often moved up in the Component tree.



https://www.geeksforgeeks.org/unidirectional-data-flow/

	props	state
Can get initial value from parent Component?	Yes	Yes
Can be changed by parent Component?	Yes	No
Can set default values inside Component?*	Yes	Yes
Can change inside Component?	No	Yes
Can set initial value for child Components?	Yes	Yes
Can change in child Components?	Yes	No

* Note that both *props* and *state* initial values received from parents override default values defined inside a Component.

https://github.com/uberVU/react-guide/blob/master/props-vs-state.md

props.children

props.children allows you to inject components inside other components.

Without props.children all extra inside component will not be displayed.

```
const Person = props => {
  return (
    <div className="box">
      <h1>{props.name}</h1>
      <div>{props.children}</div>
    </div>
  );
};
```

```
<Person name="Maria">
  <p>Some more content</p>
  <Comments />
</Person>
```

https://reactjs.org/docs/composition-vs-inheritance.html

Passing a data from event to the child

Passing a method to the child

The state can be manipulated a child component by passing a function down as a prop:

```
const Person = props => {
  return (
    <div className="box">
      <h1>{props.name}</h1>
      <button onClick={props.click}>Click me</button>
    </div>
  );
};
```

```
<Person name="Maria" click={eventHandler} />
```

Passing arguments to event handlers

```
const testHandler = (id) => {
  console.log(id);
};
```

```
<button onClick={(e) => testHandler(id, e)}> Like </button>
```

```
<button onClick={testHandler.bind(this, id)}> Like </button>
```

https://reactjs.org/docs/handling-events.html#passing-arguments-to-event-handlers

Forms and controlled components

React forms handle user input, allowing for dynamic and interactive data collection within a React application. They maintain input state, handle changes through events, and submit data. React offers controlled components for managing form elements, ensuring the React state is the single source of truth for all inputs.

Controlled components in React are form elements like <input>, <textarea>, and <select> whose values are controlled by the state within a React component. Instead of letting the DOM manage the form data, in controlled components, React is in charge. This means the form element's value is dictated by the state, and any change to the input updates the state, ensuring the React state and the form field are always in sync.

```
function SingleInputForm() {
  const [inputValue, setInputValue] = useState('');

  const handleChange = (event) => {
    setInputValue(event.target.value);
  };

  const handleSubmit = (event) => {
    event.preventDefault();
    alert('A name was submitted: ' + inputValue); };

  return (
    <form onSubmit={handleSubmit}>
      <label>
        Name:
        <input type="text" value={inputValue} onChange={handleChange} />
      </label>
      <button type="submit">Submit</button>
    </form>
  );
}

export default SingleInputForm;
```

```
function MultipleInputForm() {
  const [formState, setFormState] = useState({
    name: '',
    age: ''
  });

  const handleChange = (event) => {
    const { name, value } = event.target;
    setFormState(prevState => ({
      ...prevState,
      [name]: value
    }));
  };

  const handleSubmit = (event) => {
    event.preventDefault();
    alert(`Name: ${formState.name},
           Age: ${formState.age}`);
  };

  return (
    <form onSubmit={handleSubmit}>
      <label>
        Name:
        <input
          type="text"
          name="name"
          value={formState.name}
          onChange={handleChange}
        />
      </label>
      <label>
        Age:
        <input
          type="number"
          name="age"
          value={formState.age}
          onChange={handleChange}
        />
      </label>
      <button type="submit">Submit</button>
    </form>
  );
}
```