JSX Expressions

```
const name = 'Jack';
const element = <p>Hi, {name}</p>;

function name(firstName, lastName) {
  return firstName + ' ' + lastName;
}
const element = <p>Hello, {name('Jack','Fox')}</p>
```

JSX attributes

```
const elem = <img src={user.avatarUrl} />;
const elem = <button className="btn">Button</button>;
```

JSX Functions

```
name() {
  return "Jack";
}

return (
  <h1>Hi {name()}</h1>
)
```

JSX Conditional rendering

```
export default function Weather(props) {
  if (props.temperature >= 20) {
    return (
      <p>Warm: {props.temp}°C in {props.city}</p>
    );
  } else {
    return (
      <p>Cold: {props.temp}°C in {props.city}</p>
    );
  }
}

<Weather city="York" temp={23} />
```

Advanced functional component

```
export default function Hello(props) {
  function fullName() {
    return `${props.firstName} ${props.lastName}`;
  }
  return (
    <p>{fullName()}</p>
  );
}

<Hello firstName="Jack" lastName="Fox" />
```

Event listener

```
export default function Hello() {
  function handleClick(event) {
    event.preventDefault();
    alert("Hello");
  }

  return (
    <a onClick={handleClick}>Hi</a>
  );
}
```

React forms

```
import { useState } from "react";
export default function LoginForm() {
  const [formData, setFormData] = useState({
    username: "",
    password: "",
  });
  const handleInputChange = (e) => {
    const { name, value } = e.target;
    setFormData({
      ...formData,
      [name]: value,
    });
  };
  const handleSubmit = (e) => {
    e.preventDefault();
    console.log("Submitted data:", formData);
  };
  return (
    <form onSubmit={handleSubmit}>
      <input
        type="text" name="username"
        value={formData.username}
        onChange={handleInputChange}
      />
      <input
        type="password" name="password"
        value={formData.password}
        onChange={handleInputChange}
      />
      <button type="submit">Submit</button>
      {formData.username} {formData.password}
    </form>
  );
}
```

React PostList:

```
const PostList = ({ posts }) => {
  return (
    <div>
      {posts.map((post, index) => (
        <div key={index}>
          <h3>{post.title}</h3>
          <div>{post.body}</div>
        </div>
      ))}
    </div>
  );
};
export default PostList;

// Usage
import { useState, useEffect } from 'react';
import axios from 'axios';
const App = () => {
  const [posts, setPosts] = useState([]);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    axios.get('https://site.com/posts')
      .then(response => {
        setPosts(response.data);
        setLoading(false);
      })
      .catch(error => {
        console.error('Error:', error);
        setLoading(false);
      });
    /*fetch('https://site.com/posts')
      .then(response => response.json())
      .then(data => setPosts(data))
      .catch(error => console.error(error));*/
  }, []);

  return (
    <div>{loading ? <p>Loading...</p> : <PostList
posts={posts} />}</div>
  );
};
export default App;
```

React reducer:

```
import { useReducer } from 'react';
// Reducer function
const counterReducer = (state, action) => {
   switch (action.type) {
    case 'INCREMENT':
      return { count: state.count + 1 };
    case 'DECREMENT':
      return { count: state.count - 1 };
    default:
      return state;
  }
};
// Counter component
const Counter = ({ dispatch }) => {
  return (
    <div>
      <p>Count: {state.count}</p>
      <button onClick={() => dispatch({ type:
'INCREMENT' })}>Increment</button>
      <button onClick={() => dispatch({ type:
'DECREMENT' })}>Decrement</button>
    </div>
  );
};
// Parent component managing state
const CounterApp = () => {
  const [state, dispatch] =
useReducer(counterReducer, { count: 0 });
  return (
    <Counter dispatch={dispatch} />
  );
};
export default CounterApp;
```

useEffect(): for optimizing performance.

```
import { useState, useEffect } from 'react';
const PerformanceComponent = ({ propValue }) => {
  const [stateValue, setStateValue] = useState('');

  useEffect(() => {
    const result = doExpensiveComputation(propValue);
    setStateValue(result);
  }, [propValue]); // Re-run if propValue changes

  return (
    <div>{stateValue}</div>
  );
};
export default PerformanceComponent;
```

useEffect(): TimerComponent.

```
import { useState, useEffect } from 'react';
const TimerComponent = () => {
  const [count, setCount] = useState(0);

  useEffect(() => {
    const interval = setInterval(() => {
      setCount(prevCount => prevCount + 1);
    }, 1000);

    return () => {
      clearInterval(interval);
    };
  }, []);

  return (
    <p>Count: {count}</p>
  );
};
export default TimerComponent;
```

React Redux for state management:

Redux actions:

```
export const increment = () => ({
  type: 'INCREMENT',
});
export const decrement = () => ({
  type: 'DECREMENT',
});
```

Redux reducer:

```
const counterReducer = (state = {count:0}, action) => {
  switch (action.type) {
    case 'INCREMENT':
      return { count: state.count + 1 };
    case 'DECREMENT':
      return { count: state.count - 1 };
    default:
      return state;
  }
};
export default counterReducer;
```

Redux store:

```
import { createStore } from 'redux';
import counterReducer from './reducer';
const store = createStore(counterReducer);
export default store;
```

Counter component:

```
import { useDispatch, useSelector } from
'react-redux';
import { increment, decrement } from './actions';
const Counter = () => {
  const count = useSelector((state) =>
state.count);
  const dispatch = useDispatch();
  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() =>
dispatch(increment())}>Increment</button>
      <button onClick={() =>
dispatch(decrement())}>Decrement</button>
    </div>
  );
};
export default Counter;
```

Main App component:

```
import { Provider } from 'react-redux';
import Counter from './Counter';
import store from './store';
const App = () => {
  return (
    <Provider store={store}>
      <Counter />
    </Provider>
  );
};
export default App;
```