React v19 Features and Enhancements

1. Performance Improvements

React v19 introduces several performance optimizations, making rendering and state updates more efficient. These improvements include:

- **Improved Hydration:** Faster hydration process when rendering server-side components.
- **Optimized Reconciliation:** Reduced unnecessary updates and improved diffing algorithm.
- **Better Handling of Large Lists:** Virtualization techniques enhance performance when dealing with large data sets.

Use Case:

Performance optimizations benefit applications with dynamic content updates, reducing lag and improving responsiveness.

Example: Optimized Rendering

```
import { useState } from "react";

function List({ items }) {
  return items.map((item) => <div key={item.id}>{item.name}</div>);
}

export default function App() {
  const [items, setItems] = useState(Array.from({ length: 1000 }, (_, i) => ({ id: i, name:
`Item ${i}` })));
  return <List items={items} />;
}
```

2. Actions in React v19

Actions provide a declarative way to handle form submissions, replacing manual event handlers.

Use Case:

Ideal for handling form submissions in a structured manner, especially in server components and stateful forms.

Example: Handling Actions

```
import { useActionState } from "react";

async function submitData(formData) {
  return new Promise((resolve) => setTimeout(() => resolve("Submitted: " +
formData.get("name")), 1000));
}

export default function FormComponent() {
  const [state, formAction] = useActionState(async (prev, formData) => await
submitData(formData), null);

  return (
    <form action={formAction}>
      <input name="name" required />
      <button type="submit">Submit</button>
      {state && <p>{state}</p>}
    </form>
  );
}
```

3. useActionState Hook

This hook enables state management inside actions, providing a way to track form submission results.

Use Case:

Useful for handling async form submissions without requiring manual state management.

Example:

```
const [state, action] = useActionState(async (prev, formData) => {
  return `Processed: ${formData.get("data")}`;
}, null);
```

4. Handling <form> Actions in React DOM

Forms now support action attributes that streamline submission handling.

Use Case:

Great for reducing boilerplate code and handling form logic in a more declarative way.

Example:

```
<form action={async (formData) => console.log("Form Submitted: ", formData)}>
  <input name="email" type="email" required />
  <button type="submit">Submit</button>
</form>
```

5. useFormStatus Hook

This hook provides real-time form submission status, enabling UI feedback during submission.

Use Case:

Helpful in disabling submit buttons and showing loading indicators.

Example:

```
import { useFormStatus } from "react";

function SubmitButton() {
  const { pending } = useFormStatus();
  return <button type="submit" disabled={pending}>{pending ? "Submitting..." :
"Submit"}</button>;
}
```

6. useOptimistic Hook

This hook provides optimistic UI updates, making user interactions feel instant.

Use Case:

Improves perceived performance by updating the UI before receiving a server response.

Example:

```
import { useOptimistic } from "react";

export default function OptimisticCounter() {
  const [count, setCount] = useOptimistic(0);
  return (
    <button onClick={() => setCount((c) => c + 1)}>
      Count: {count}
    </button>
  );
}
```

7. New use API

The **use** API allows direct usage of async data within components.

Use Case:

Simplifies async data fetching and prevents unnecessary re-renders.

Example:

```
import { use } from "react";

async function fetchData() {
  return new Promise((resolve) => setTimeout(() => resolve("Fetched Data"), 1000));
}

export default function DataComponent() {
  const data = use(fetchData());
  return <p>{data}</p>;
}
```

8. useDeferredValue Hook and Initial Value Update in v19

This hook defers updates to a state value, improving performance.

Use Case:

Useful for delaying non-essential updates, such as filtering large lists.

Example:

```
import { useDeferredValue, useState } from "react";

export default function DeferredComponent() {
  const [value, setValue] = useState("");
  const deferredValue = useDeferredValue(value);
  return (
    <div>
      <input onChange={(e) => setValue(e.target.value)} />
      <p>Deferred: {deferredValue}</p>
    </div>
  );
}
```

9. ref as a Prop in v19

React v19 allows passing **ref** as a prop, making component composition easier.

Use Case:

Improves usability for components that need to expose DOM elements.

Example:

```
import { useRef } from "react";

function InputComponent({ refProp }) {
  return <input ref={refProp} />;
}

export default function App() {
  const inputRef = useRef(null);
  return <InputComponent refProp={inputRef} />;
}
```

10. Context as a Provider

React v19 allows using context directly as a provider component.

Use Case:

Simplifies state sharing across deeply nested components.

Example:

```
import { createContext } from "react";

const ThemeContext = createContext("");

function App({ children }) {
  return (
    <ThemeContext value="dark">
      {children}
    </ThemeContext>
  );
}
```

11. Creating a Project with Vite

Vite is a fast-build tool for React applications.

Steps to Create a React App with Vite:

➔ Install Vite:

```
npm create vite@latest my-react-app --template react
```

➔ Navigate to the project:

```
cd my-react-app
```

➔ Install dependencies:

```
npm install
```

➔ Start the development server:

```
npm run dev
```

Conclusion:

React v19 introduces significant improvements in performance, form handling, and state management. Features like useActionState, useFormStatus, useOptimistic, and useDeferredValue make React applications more efficient and responsive. The new ‘use’ API and direct context provider usage simplifies development, reducing boilerplate code. Overall, these enhancements ensure a better developer experience and smoother user interactions.

References:

- https://react.dev/blog/2024/12/05/react-19
- https://react.dev/reference/react/useActionState
- https://react.dev/reference/react-dom/hooks/useFormStatus
- https://react.dev/reference/react/useOptimistic
- https://react.dev/reference/react/use
- https://react.dev/reference/react/useDeferredValue