Berikut adalah penjelasan README untuk kode React di atas yang dapat membantu kamu memahami setiap bagian kode tersebut, dengan penjelasan yang disesuaikan agar mudah dipahami oleh pemula.

Panduan Pemahaman Kode React untuk Aplikasi Quiz

Pengertian Umum

Pada kode ini, kita membuat aplikasi **quiz** (kuis) menggunakan React. Aplikasi ini mengambil data pertanyaan kuis dari server, memproses jawaban, dan menghitung poin berdasarkan jawaban yang diberikan.

Secara keseluruhan, aplikasi ini mengelola status kuis (seperti pertanyaan mana yang sedang dijawab, poin yang diperoleh, sisa waktu, dll.) menggunakan **React Context** dan **useReducer** untuk manajemen state.

Cara Berpikir React

Pada React, data dan logika aplikasi biasanya dikelola dalam komponen yang terpisah, dan data tersebut disalurkan melalui **Context API** untuk bisa diakses di seluruh aplikasi. Untuk mengelola perubahan data, kita menggunakan **reducer**, yang bekerja mirip dengan pengelolaan status di aplikasi desktop atau mobile.

Berikut ini adalah cara berpikir aplikasi kuis ini:

1. **Data Pertanyaan:** Mengambil data pertanyaan dari server dan menyimpannya dalam state.
2. **Status Aplikasi:** Mengelola status aplikasi (apakah sedang loading, sedang aktif, selesai, atau error).
3. **Interaksi Pengguna:** Mengelola interaksi pengguna (menjawab soal, berpindah ke soal berikutnya, menghitung poin).
4. **Waktu:** Menghitung waktu yang tersisa untuk setiap soal.

Analogi Sederhana

Bayangkan kamu bermain kuis dengan teman-teman. Setiap soal memiliki waktu untuk dijawab. Jika kamu menjawab benar, kamu mendapatkan poin. Namun, jika waktu habis atau jawaban salah, kuis berakhir atau berpindah ke soal berikutnya. Semua data ini (soal, poin, waktu, status kuis) harus dikelola dengan baik agar permainan berjalan lancar.

Penjelasan Kode Tiap Baris

Berikut adalah penjelasan untuk kode yang diberikan, yang menggunakan **React Context**, **useReducer**, dan **useEffect**.

1. Mengimpor Library dan Membuat Context

```
import { createContext, useContext, useEffect, useReducer } from "react";
```

- `createContext` digunakan untuk membuat **Context** yang akan menyediakan data kuis (seperti status, soal, poin, dll.) kepada komponen lain di aplikasi.
- `useContext` digunakan untuk mengakses data yang disediakan oleh `QuizContext`.
- `useEffect` digunakan untuk melakukan **side-effect** (seperti mengambil data dari server) saat komponen pertama kali di-render.
- `useReducer` digunakan untuk mengelola state aplikasi dengan lebih kompleks, seperti dalam kasus kuis ini, di mana banyak aksi yang mempengaruhi state.

2. Membuat Context dan Variabel Konstanta

```
const QuizContext = createContext();
const SECS_PER_QUESTION = 30;
```

- `QuizContext` adalah konteks yang akan digunakan untuk mengakses data kuis di seluruh aplikasi.
- `SECS_PER_QUESTION` adalah jumlah detik yang diberikan untuk setiap soal dalam kuis.

3. Mendefinisikan `initialState`

```
const initialState = {
  questions: [],
  status: "loading", // status aplikasi, bisa "loading", "error", "ready",
"active", atau "finished"
  index: 0, // indeks soal yang sedang dijawab
  answer: null, // jawaban yang dipilih oleh pengguna
  points: 0, // poin yang diperoleh
  highscore: 0, // skor tertinggi yang pernah diperoleh
  secondRemaining: null, // waktu yang tersisa untuk soal saat ini
};
```

- `initialState` adalah state awal yang akan digunakan saat aplikasi pertama kali dimulai. Ini berisi data tentang soal kuis, status aplikasi, indeks soal yang sedang dijawab, jawaban, poin, dan waktu tersisa.

4. Fungsi `reducer` untuk Mengelola State

```
function reducer(state, action) {
  switch (action.type) {
    case "dataReceived":
      return {
        ...state,
        questions: action.payload,
        status: "ready",
      };
    case "dataFailed":
      return {
        ...state,
```

```
        status: "error",
      };
    case "start":
      return {
        ...state,
        status: "active",
        secondRemaining: state.questions.length * SECS_PER_QUESTION,
      };
    case "newAnswer": {
      const question = state.questions.at(state.index);

      return {
        ...state,
        answer: action.payload,
        points:
          action.payload === question.correctOption
            ? state.points + question.points
            : state.points,
      };
    }
    case "nextQuestion":
      return {
        ...state,
        index: state.index + 1,
        answer: null,
      };
    case "finish":
      return {
        ...state,
        status: "finished",
        highscore:
          state.points > state.highscore ? state.points : state.highscore,
      };
    case "restart":
      return {
        ...initialState,
        questions: state.questions,
        status: "ready",
      };
    case "tick":
      return {
        ...state,
        secondRemaining: state.secondRemaining - 1,
        status: state.secondRemaining === 0 ? "finished" : state.status,
      };
    default:
      throw new Error("Action unknown");
  }
}
```

- `reducer` adalah fungsi yang mengubah state aplikasi berdasarkan **action** yang diberikan. Ada berbagai jenis **action** yang bisa mengubah state:

- `dataReceived`: Ketika data pertanyaan berhasil diterima, status berubah menjadi "ready".
- `dataFailed`: Ketika pengambilan data gagal, status berubah menjadi "error".
- `start`: Mengubah status menjadi "active" dan mengatur waktu untuk setiap soal.
- `newAnswer`: Memproses jawaban yang diberikan dan menghitung poin jika jawaban benar.
- `nextQuestion`: Berpindah ke soal berikutnya.
- `finish`: Mengakhiri kuis dan menyimpan highscore jika poin lebih tinggi dari sebelumnya.
- `restart`: Mengulang kuis dari awal.
- `tick`: Menurunkan waktu yang tersisa untuk soal yang sedang dijawab.

5. `QuizProvider` untuk Menyediakan Context

```
function QuizProvider({ children }) {
  const [
    { questions, status, index, answer, points, highscore, secondRemaining
},
    dispatch,
  ] = useReducer(reducer, initialState);
```

- `useReducer` digunakan di sini untuk mengelola state kuis. Kita mendapatkan state dan fungsi `dispatch` untuk mengubah state. `dispatch` akan digunakan untuk memicu aksi-aksi seperti mulai kuis, pindah soal, atau memperbarui waktu yang tersisa.

```
const numQuestions = questions.length;
const maxPoissiblePoints = questions.reduce(
  (prev, cur) => prev + cur.points,
  0
);
```

- `numQuestions` adalah jumlah total soal.
- `maxPoissiblePoints` adalah jumlah total poin yang bisa diperoleh dari semua soal.

6. Mengambil Data dan Menyediakan Context

```
useEffect(() => {
  const fetchData = async () => {
    try {
      const res = await fetch("http://localhost:8000/questions");
      if (!res.ok) {
        throw new Error("Data fetch failed");
      }
      const data = await res.json();
      dispatch({ type: "dataReceived", payload: data });
    } catch (err) {
      console.error("Error fetching data:", err);
      dispatch({ type: "dataFailed", error: err.message });
    }
```

```
    };

    fetchData();
  }, []);
```

- `useEffect` digunakan untuk mengambil data dari server saat komponen pertama kali dimuat. Jika berhasil, data akan disimpan di state melalui aksi `dataReceived`. Jika gagal, aksi `dataFailed` akan dipicu.

```
  return (
    <QuizContext.Provider
      value={{
        questions,
        status,
        index,
        answer,
        points,
        highscore,
        secondRemaining,
        numQuestions,
        maxPoissiblePoints,
        dispatch,
      }}
    >
      {children}
    </QuizContext.Provider>
  );
}
```

- `QuizContext.Provider` menyediakan data kuis kepada semua komponen yang berada di dalamnya melalui **Context**. Data yang disediakan meliputi soal, status kuis, poin, waktu tersisa, dan fungsi `dispatch`.

7. Hook `useQuiz` untuk Mengakses Data

```
function useQuiz() {
  const context = useContext(QuizContext);
  if (context === undefined)
    throw new Error("QuizContext was used outside the QuizProvider");
  return context;
}
```

- `useQuiz` adalah hook kustom untuk mengakses data kuis yang disediakan oleh `QuizContext`. Jika digunakan di luar `QuizProvider`, maka akan menghasilkan error.

Kesimpulan

Dengan kode ini, kita telah membuat aplikasi kuis yang

dikelola dengan baik menggunakan **React Context**, **useReducer**, dan **useEffect**. Konsep utama di sini adalah:

- **State management** dengan `useReducer` untuk menangani berbagai aksi dan status kuis.
- **Context API** untuk membagikan data kuis ke seluruh aplikasi.
- `useEffect` untuk mengambil data dari server saat aplikasi dimulai.

Kamu bisa menggunakan `QuizProvider` untuk membungkus aplikasi kamu dan `useQuiz` untuk mengakses data kuis di bagian lain aplikasi.