1. ¿Qué es React Context?

React Context es una característica de React que permite compartir datos entre componentes sin tener que pasar props manualmente a través de cada nivel del árbol de componentes. Proporciona una forma de crear un contexto global en la aplicación, donde los datos pueden ser consumidos y actualizados desde cualquier componente dentro del árbol.

2. ¿Cómo se utiliza React Context?

El uso de React Context implica tres pasos principales: la creación del contexto, la provisión de datos y el consumo de datos.

3. Creación del contexto:

Primero, se crea el contexto utilizando el método React.createContext(). Esto devuelve un objeto que contiene dos componentes: Proveedor (Provider) y Consumidor (Consumer). El contexto creado se utiliza para envolver los componentes que necesitan acceder a los datos compartidos.

```
const MiContexto = React.createContext();
```

3.1. Provisión de datos:

Se utiliza el componente Proveedor (Provider) para envolver los componentes que necesitan acceder a los datos compartidos. El componente Proveedor acepta una prop value que contiene los datos que se compartirán.

```
<MiContexto.Provider value={datosCompartidos}>
  {/* Componentes que necesitan acceder a los datos compartidos */}
</MiContexto.Provider>
```

3.2. Consumo de datos:

Dentro de los componentes que necesitan acceder a los datos compartidos, se utiliza el componente Consumidor (Consumer) para acceder a esos datos. Esto se puede hacer utilizando una función como hijo (child function) o el hook useContext.

3.2.1. Utilizando una función como hijo:

```
<MiContexto.Consumer>
  {datos => (
    {/* Utilizar los datos compartidos */}
  )}
</MiContexto.Consumer>
```

3.2.2. Utilizando el hook useContext:

```
const datos = React.useContext(MiContexto);
```

4. Ejemplos de React Context:

4.1. Ejemplo 1: Tema de la aplicación

En este ejemplo, se utiliza React Context para compartir el tema de la aplicación entre múltiples componentes.

```
const TemaContexto = React.createContext();

const ProveedorTema = ({ children }) => {
  const tema = "dark";

  return (
    <TemaContexto.Provider value={tema}>
      {children}
    </TemaContexto.Provider>
  );
};

const ComponenteConsumidor = () => {
  const tema = React.useContext(TemaContexto);

  return <p>Tema actual: {tema}</p>;
};
```

```
const App = () => (
  <ProveedorTema>
    <ComponenteConsumidor />
  </ProveedorTema>
);
```

4.2. Ejemplo 2: Autenticación del usuario

En este ejemplo, se utiliza React Context para compartir la información de autenticación del usuario en la aplicación.

```
const UsuarioContexto = React.createContext();

const ProveedorUsuario = ({ children }) => {
  const usuario = {
    nombre: "John Doe",
    email: "johndoe@example.com",
    isLoggedIn: true,
  };

  return (
    <UsuarioContexto.Provider value={usuario}>
      {children}
    </UsuarioContexto.Provider>
  );
};

const ComponenteConsumidor = () => {
  const usuario = React.useContext(UsuarioContexto);

  return (
```

```
    <div>
      <p>Nombre: {usuario.nombre}</p>
      <p>Email: {usuario.email}</p>
      <p>¿Está logueado? {usuario.isLoggedIn ? "Sí" : "No"}</p>
    </div>
  );
};

const App = () => (
  <ProveedorUsuario>
    <ComponenteConsumidor />
  </ProveedorUsuario>
);
```

Estos ejemplos ilustran cómo se crea un contexto, cómo se proveen los datos y cómo se consumen en los componentes necesarios. React Context es una poderosa herramienta que permite compartir datos globalmente en una aplicación de React, evitando la necesidad de pasar props a través de múltiples niveles de componentes.