### 1. Componente `Display.js`

```jsx
// Este componente se encargará de mostrar el resultado de la operación.

import React from 'react';

export const Display = ({ value }) => (
  <div className="display">
    {value}
  </div>
);

export default Display;
```

- **`import React from 'react';`**:
  - Importa la biblioteca React, que es esencial para definir componentes en JSX. Aunque en versiones recientes de React no es estrictamente necesario importar React para JSX, es una práctica común y necesaria en algunas configuraciones.
- **`const Display = ({ value }) => (`**:
  - Define un componente funcional llamado `Display`.
  - Utiliza la desestructuración de objetos para extraer la prop `value` directamente de las props del componente.
- **`<div className="display">`**:
  - Retorna un elemento `div` con la clase CSS `display`. Esta clase se utilizará para aplicar estilos específicos al contenedor del valor.
- **`{value}`**:
  - Inserta el valor de la prop `value` dentro del `div`. Este es el valor que se mostrará en la pantalla de la calculadora.
- **`</div>`**:
  - Cierra el elemento `div`.
- **`);`**:
  - Cierra la declaración del componente funcional.
- **`export default Display;`**:
  - Exporta el componente `Display` como exportación por defecto, permitiendo que sea importado en otros archivos.

**2. Componente `Botones.js`**

```jsx
import React from 'react';
import './Botones.css';

const Botones = ({ onButtonClick }) => {
  const buttons = [
    'C', '+/-', '%', '/',
    '7', '8', '9', '*',
    '4', '5', '6', '-',
    '1', '2', '3', '+',
    '0', '.', '='
  ];

  return (
    <div className="botones">
      {buttons.map((btn) => (
        <button
          key={btn}
          onClick={() => onButtonClick(btn)}
          className={btn === '0' ? 'button double' : 'button'}
        >
          {btn}
        </button>
      ))}
    </div>
  );
};

export default Botones;
```

- **`import React from 'react';`**:
  - Importa la biblioteca React, necesaria para definir componentes en JSX.
- **`import './Botones.css';`**:
  - Importa el archivo de estilos `Botones.css`, que contiene los estilos específicos para los botones de la calculadora.
- **`const Botones = ({ onButtonClick }) => {`**:
  - Define un componente funcional llamado `Botones`.
  - Utiliza la desestructuración de objetos para extraer la prop `onButtonClick` directamente de las props del componente.
- **`const buttons = [`**:
  - Declara un arreglo llamado `buttons` que contiene los valores de los botones de la calculadora.
- **`'C', '+/-', '%', '/',`**:

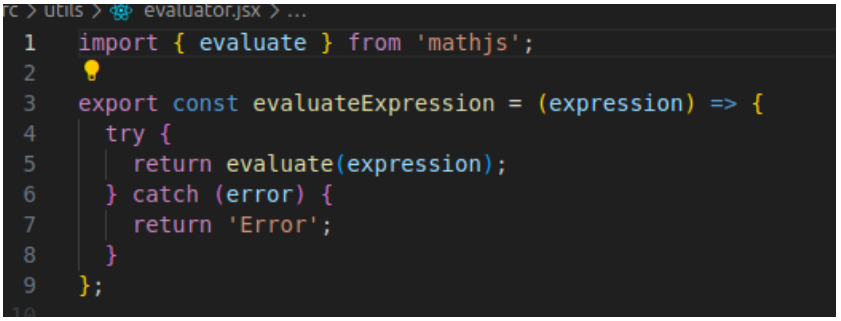
    - Elementos del arreglo `buttons` que representan las operaciones y funciones especiales de la calculadora.
- `'7', '8', '9', '*',`:
    - Números y operaciones adicionales en el arreglo `buttons`.
- `'4', '5', '6', '-',`:
    - Más números y operaciones en el arreglo `buttons`.
- `'1', '2', '3', '+',`:
    - Continuación de números y operaciones en el arreglo `buttons`.
- `'0', '.', '='`:
    - Últimos elementos del arreglo `buttons`, incluyendo el cero, el punto decimal y el signo igual.
- `];`:
    - Cierra la declaración del arreglo `buttons`.
- `return (`:
    - Inicia el retorno del JSX que define la estructura del componente.
- `<div className="button-container">`:
    - Retorna un elemento `div` con la clase CSS `button-container`, que se utilizará para aplicar estilos al contenedor de los botones.
- `{buttons.map((btn) => (`:
    - **Qué hace:** Aquí estamos usando la función `map` de JavaScript para recorrer el array `buttons` y crear un nuevo componente `<button>` para cada valor en el array.

- `<button key={btn} onClick={() => onButtonClick(btn)} className={btn === '0' ? 'button double' : 'button'}>`
    - **Qué hace:** Por cada valor de `btn`, se genera un elemento `<button>` en el DOM. Este botón tiene tres propiedades principales:
    - `key={btn}`:
        - **Qué hace:** Asigna un `key` único a cada botón, basado en el valor de `btn`. `key` es una propiedad especial en React que ayuda a React a identificar qué elementos han cambiado, se han agregado o eliminado. En este caso, el valor del botón (`btn`) se utiliza como clave.
        - **Por qué es importante:** En React, cuando se renderiza una lista de elementos (como los botones en este caso), se necesita una clave única para cada uno. Esto ayuda a React a hacer un seguimiento eficiente de los elementos y mejorar el rendimiento.
    - `onClick={() => onButtonClick(btn)}`:
        - **Qué hace:** Asocia un evento de clic a cada botón. Cuando el usuario hace clic en un botón, se ejecuta la función `onButtonClick`, pasando el valor de `btn` como argumento.

        - **Por qué es importante:** Esto permite que, cuando se hace clic en un botón, se realice una acción en función del valor del botón (por ejemplo, realizar una operación en una calculadora).
    - `className={btn === '0' ? 'button double' : 'button'}`:
        - **Qué hace:** Asigna una clase CSS al botón. Si el valor de `btn` es `'0'`, se asigna la clase `button double`. En caso contrario, se asigna solo la clase `button`.
        - **Por qué es importante:** Esto se hace para aplicar un estilo especial a los botones. Por ejemplo, si el valor de `btn` es `'0'`, se le da la clase `double`, que probablemente hace que el botón de `'0'` ocupe más espacio (más ancho) en la interfaz (como en una calculadora). Los demás botones solo tendrán la clase `button`, que probablemente tenga un estilo por defecto.

- `onClick={() => onButtonClick(btn)}`:
    - Asigna un manejador de eventos `onClick` a cada botón. Cuando se hace clic en un botón, se ejecuta la función `onButtonClick` pasada

### 3. Utils `evaluator.js`




```
import { evaluate } from 'mathjs';
```

- **Qué hace:** Esta línea importa la función `evaluate` desde la librería `mathjs`.
- **Por qué es necesario:** `mathjs` es una librería de matemáticas que permite realizar cálculos y evaluaciones de expresiones matemáticas de manera sencilla. La función `evaluate` es utilizada para analizar y calcular el resultado de una expresión matemática que se pasa como un string.

```
export const evaluateExpression = (expression) => {
```

- **Qué hace:** Define una función llamada `evaluateExpression`, que toma un parámetro de entrada llamado `expression`.

- **Por qué es necesario:** `evaluateExpression` es la función que se exporta, lo que significa que estará disponible para ser utilizada en otros archivos del proyecto. El parámetro `expression` es una cadena de texto que representa la expresión matemática que se desea evaluar (por ejemplo, `'2+3*4'`).

```
return evaluate(expression);
```

- **Qué hace:** Llama a la función `evaluate` de `mathjs`, pasándole la `expression` como argumento.
- **Por qué es necesario:** La función `evaluate` intenta calcular el valor de la expresión matemática que se pasa como cadena de texto. Si la expresión es válida, devuelve el resultado del cálculo. Por ejemplo, `'2+3*4'` devolvería `14`.
- **Qué hace:** Si ocurre un error durante la evaluación de la expresión, esta línea devuelve el string `'Error'`.
- **Por qué es necesario:** Esto previene que el programa falle o se detenga si la expresión es inválida. En lugar de lanzar un error, se devuelve un mensaje de error amigable (`'Error'`), lo que facilita la gestión del problema.

### 4. Componente `App.js`

```jsx
import React, { useState } from 'react';
import Display from './components/Display';
import Botones from './components/Botones';
import { evaluateExpression } from './utils/evaluator';
import './App.css';

const App = () => {
  const [expression, setExpression] = useState('');

  const handleButtonClick = (btn) => {
    if (btn === 'C') {
      setExpression('');
    } else if (btn === '=') {
      const result = evaluateExpression(expression);
      setExpression(result.toString());
    } else if (btn === '+/-') {
      setExpression((prev) => (prev.charAt(0) === '-' ? prev.slice(1) : `-${prev}`));
    } else {
      setExpression((prev) => prev + btn);
    }
  };

  return (
    <div className="app">
      <h1>Calculadora</h1>
      <Display value={expression || '0'} />
      <Botones onButtonClick={handleButtonClick} />
    </div>
  );
};

export default App;
```

- **`import React, { useState } from 'react';`**:
  - Importa la biblioteca React y el hook `useState` desde React. `useState` se utiliza para manejar el estado en componentes funcionales.
- **`import Display from './components/Display';`**:
  - Importa el componente `Display` desde el archivo `Display.js` ubicado en la carpeta `components`. Este componente se encargará de mostrar la expresión matemática actual.
- **`import Botones from './components/Botones';`**:
  - Importa el componente `Botones` desde el archivo `Botones.js` en la misma carpeta. Este componente contiene los botones de la calculadora y maneja las interacciones del usuario.
- **`import { evaluateExpression } from './utils/evaluator';`**:
  - Importa la función `evaluateExpression` desde el archivo `evaluator.js` en la carpeta `utils`. Esta función se encargará de evaluar la expresión matemática ingresada por el usuario.

- `import './App.css';`:
    - Importa el archivo de estilos `App.css` para aplicar estilos específicos al componente `App`.
- `const App = () => {`:
    - Define el componente funcional `App`.
- `const [expression, setExpression] = useState('');`:
    - Declara una variable de estado llamada `expression` y su función de actualización `setExpression`. Inicializa `expression` como una cadena vacía. Esta variable almacenará la expresión matemática actual.
    - Si el botón presionado no es ninguno de los anteriores, se asume que es un

```
const handleButtonClick = (btn) => {
```

- **Qué hace:** Esta línea define una función llamada `handleButtonClick`. Esta función recibe un argumento llamado `btn`, que es el valor del botón que se hizo clic. Este valor puede ser un número, un operador matemático, o una operación especial como "C" (limpiar), "=" (evaluar) o "+/-" (invertir el signo).
- **Por qué es necesario:** `btn` es la entrada que nos dice qué botón ha sido presionado para realizar la acción correspondiente.

```
if (btn === 'C') return setExpression('');
```

- **Qué hace:** Compara si el valor de `btn` es igual a `'C'` (lo que normalmente significa "limpiar").
- **Por qué es necesario:** Si el botón presionado es `'C'`, la expresión se limpia y se reinicia a una cadena vacía. Esto hace que la pantalla de la calculadora se vacíe cuando se presiona el botón "C".
- `setExpression('')`: Esta línea llama a la función `setExpression` para actualizar el estado de la expresión con un valor vacío, eliminando todo lo que haya en la pantalla.

```
if (btn === '=') return
setExpression(evaluateExpression(expresion).toString());
```

- **Qué hace:** Compara si el valor de `btn` es igual a `'='` (lo que generalmente significa "evaluar la expresión").
- **Por qué es necesario:** Si el botón presionado es `'='`, se evalúa la expresión actual para obtener el resultado.
- `evaluateExpression(expresion)`: Llama a la función `evaluateExpression` para calcular el resultado de la expresión (asumimos que `expresion` contiene el texto de la operación matemática).
- `toString()`: Convierte el resultado en un texto para que se pueda mostrar en la pantalla de la calculadora.
- `setExpression()`: Actualiza el estado con el resultado de la expresión calculada.

```
if (btn === '+/-') return setExpression((prev) => prev[0] === '-' ? prev.substring(1) : '-' + prev);
```

- **Qué hace:** Compara si el valor de `btn` es igual a `'+/-'` (lo que generalmente significa "invertir el signo").
- **Por qué es necesario:** Si el botón presionado es `'+/-'`, cambia el signo de la expresión. Si la expresión comienza con un signo negativo, lo elimina; si no tiene signo negativo, lo agrega.
- **`prev[0] === '-'`**: Verifica si el primer carácter de la expresión es un signo negativo (`'-'`).
- **`prev.substring(1)`**: Si la expresión comienza con un `'-'`, se elimina ese signo negativo con `substring(1)`, que devuelve la cadena sin el primer carácter.
- **`'-' + prev`**: Si la expresión no tiene un `'-'` al principio, se agrega un signo negativo al principio de la expresión.
- **`setExpression((prev) => ...)`**: Actualiza el estado de la expresión con el nuevo valor, ya sea con el signo cambiado o añadido.

```
setExpression((prev) => prev + btn);
```

- **Qué hace:** Esta línea se ejecuta si ninguna de las condiciones anteriores se cumple, lo que significa que el botón presionado es un número o un operador matemático (por ejemplo, `1`, `+`, `*`, etc.).
- **Por qué es necesario:** En este caso, el botón presionado se añade al final de la expresión que está siendo ingresada.
- **`prev + btn`**: Agrega el valor de `btn` al final de la cadena de texto `prev` (la expresión actual). Si el botón presionado es un número o operador, este se agrega a la expresión existente.
- **`setExpression((prev) => prev + btn)`**: Actualiza el estado con la nueva expresión concatenada con el valor de `btn`.
- **`return (`**:
  - Inicia el retorno del JSX que define la estructura del componente.
- **`<div className="app">`**:
  - Retorna un elemento `div` con la clase CSS `app`, que se utilizará para aplicar estilos al contenedor principal de la aplicación.
- **`<h1>Calculadora</h1>`**:
  - Incluye un encabezado `h1` con el texto "Calculadora".
- **`<Display value={expression || '0'} />`**:

  Estamos pasando una propiedad (prop) llamada `value` al componente `Display`. El valor de `value` será el valor de la variable `expresion` si tiene algún valor, o `'0'` si `expresion` es falsy (como una cadena vacía o `null`).

- **¿Por qué se usa `|| '0'`?**: Esto es un truco para mostrar un valor predeterminado de `'0'` si la variable `expresion` está vacía o no tiene valor. Es una forma de

asegurarse de que siempre haya algo visible en la pantalla (incluso si no se ha ingresado nada).

- **`<Botones onButtonClick={handleButtonClick} />`**:
- **Qué hace:** Aquí estamos renderizando otro componente llamado `Botones`. Este componente es responsable de mostrar los botones de la calculadora (números, operadores, etc.) y manejar la interacción del usuario.
- **`onButtonClick={handleButtonClick}`**: Estamos pasando una prop llamada `onButtonClick` al componente `Botones`. El valor de esta prop es la función `handleButtonClick`, que se ejecutará cada vez que un usuario haga clic en uno de los botones de la calculadora.
    - **¿Por qué se pasa una función?**: `handleButtonClick` es la función que hemos definido anteriormente para manejar las acciones cuando un usuario hace clic en los botones de la calculadora. Esta función toma el valor del botón clickeado y decide qué acción tomar (como limpiar la pantalla, agregar un número, etc.).