Universidad Juárez del Estado de Durango

Facultad de Ingeniería Ciencias y Arquitectura

Ingeniería en Tecnologías Computacionales

Programación Web

Reporte de Práctica 25

Diego Rea Morales

Ing. Fabian Gallegos Gutiérrez

7 de noviembre de 2025

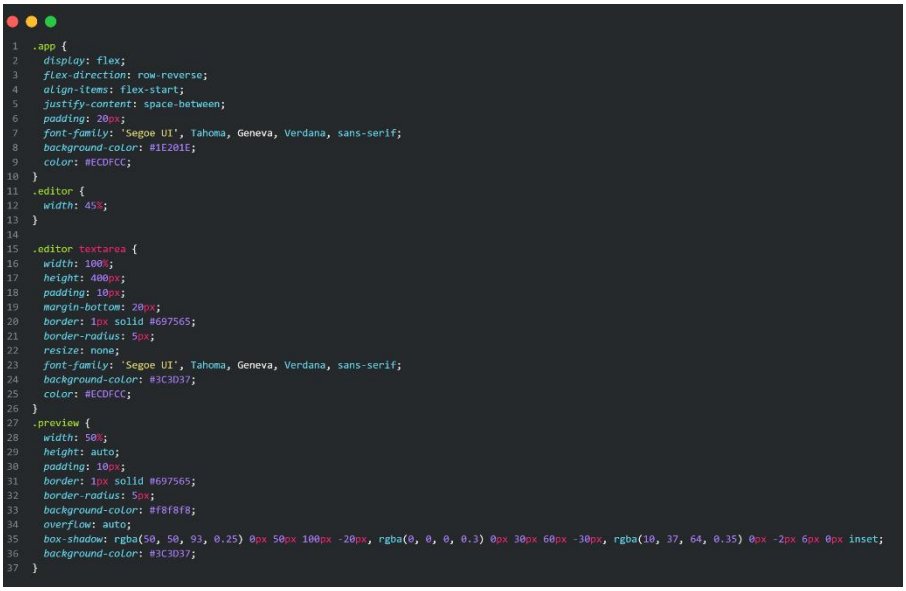
Índice

Contenido

El propósito es el de crear un editor de código HTML y CSS en tiempo real.

Es decir que el usuario cree y pruebe tanto código HTML como CSS y el resultado de este se muestre en un contenedor de la página.

```jsx
import React, { useState } from "react"; // Importa React y la función useState desde la biblioteca de React
import "./App.css"; // Importa el archivo de estilos CSS para la aplicación
function App() {
  // Define el componente función principal llamado App
  const [htmlCode, setHtmlCode] = useState(""); // Define un estado llamado htmlCode para almacenar el código HTML y una función setHtmlCode para actualizar ese estado
  const [cssCode, setCssCode] = useState(""); // Define un estado llamado cssCode para almacenar el código CSS y una función setCssCode para actualizar ese estado
  const handleHtmlChange = (event) => {
    // Define una función handleHtmlChange que se ejecuta cuando hay un cambio en el textarea de HTML
    setHtmlCode(event.target.value); // Actualiza el estado htmlCode con el valor del textarea
  };
  const handleCssChange = (event) => {
    // Define una función handleCssChange que se ejecuta cuando hay un cambio en el textarea de CSS
    setCssCode(event.target.value); // Actualiza el estado cssCode con el valor del textarea
  };
  return (
    // Devuelve la estructura del componente
    <div className="app">
      {" "}
      {/* Crea un contenedor principal con la clase 'app' */}
      <div className="editor">
        {" "}
        {/* Crea un contenedor para el editor */}
        <textarea // Crea un textarea para el código HTML
          className="html-editor" // Asigna la clase 'html-editor' al textarea
          placeholder="Escriba HTML aquí..." // Establece un marcador de posición para el textarea
          value={htmlCode} // Establece el valor del textarea como htmlCode
          onChange={handleHtmlChange} // Asigna la función handleHtmlChange al evento onChange
        ></textarea>
        <textarea // Crea un textarea para el código CSS
          className="css-editor" // Asigna la clase 'css-editor' al textarea
          placeholder="Escriba CSS aquí..." // Establece un marcador de posición para el textarea
          value={cssCode} // Establece el valor del textarea como cssCode
          onChange={handleCssChange} // Asigna la función handleCssChange al evento onChange
        ></textarea>
      </div>
      <div
        className="preview"
        dangerouslySetInnerHTML={{
          __html: `<style>${cssCode}</style>${htmlCode}`,
        }}
      />{" "}
      {/* Crea un div para mostrar la vista previa del código */}
    </div> // Cierra el contenedor principal
  );
}
export default App; // Exporta el componente App para que pueda ser importado en otros archivos
```



```css
.app {
  display: flex;
  flex-direction: row-reverse;
  align-items: flex-start;
  justify-content: space-between;
  padding: 20px;
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  background-color: #1E201E;
  color: #ECDFCC;
}
.editor {
  width: 45%;
}

.editor textarea {
  width: 100%;
  height: 400px;
  padding: 10px;
  margin-bottom: 20px;
  border: 1px solid #697565;
  border-radius: 5px;
  resize: none;
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  background-color: #3C3D37;
  color: #ECDFCC;
}
.preview {
  width: 50%;
  height: auto;
  padding: 10px;
  border: 1px solid #697565;
  border-radius: 5px;
  background-color: #f8f8f8;
  overflow: auto;
  box-shadow: rgba(50, 50, 93, 0.25) 0px 50px 100px -20px, rgba(0, 0, 0, 0.3) 0px 30px 60px -30px, rgba(10, 37, 64, 0.35) 0px -2px 6px 0px inset;
  background-color: #3C3D37;
}
```

Explicación del código JavaScript

```
const [htmlCode, setHtmlCode] = useState("");
const [cssCode, setCssCode] = useState("");
```

Primero tenemos los *hooks*, que corresponden al espacio donde el código HTML y CSS, respectivamente que el usuario ingresó.

```
const handleHtmlChange = (event) => {
    setHtmlCode(event.target.value);
};
  const handleCssChange = (event) => {
    setCssCode(event.target.value);
};
```

Después, para poder setear los valores, los eventos de cambio del valor del input.

```
<div className="app">
      {" "}
      {/* Crea un contenedor principal con la clase 'app' */}
      <div className="editor">
        {" "}
        {/* Crea un contenedor para el editor */}
        <textarea
          className="html-editor"
          placeholder="Escriba HTML aquí..."
          value={htmlCode}
          onChange={handleHtmlChange}
        ></textarea>
        <textarea
          className="css-editor"
          placeholder="Escriba CSS aquí..."
          value={cssCode}
          onChange={handleCssChange}
        ></textarea>
      </div>
      <div
        className="preview"
        dangerouslySetInnerHTML={{
          __html: `<style>${cssCode}</style>${htmlCode}`,
        }}
      />{" "}
      {/* Crea un div para mostrar la vista previa del código */}
    </div>
```

Para que después el código que el usuario ingresó se ingresará dentro de un contenedor.

Conclusiones

Para resumir, esta práctica dejó clarísimo cómo funciona la idea central de React: la vista previa que ves es simplemente un reflejo directo del código que está guardado en los estados htmlCode y cssCode. En el momento en que escribes algo en los textarea, el estado se actualiza, y la vista previa se "dibuja" sola casi al instante.