Curso de ProgramaciÃ³n Desde Cero

Tema 4: React y Desarrollo Moderno

4.1 IntroducciÃ³n a React

React es una biblioteca de JavaScript desarrollada por Facebook (ahora Meta) para construir interfaces de usuario interactivas y dinÃ¡micas. React ha revolucionado el desarrollo web frontend y es una de las tecnologÃ-as mÃ¡s demandadas en la industria tecnolÃ³gica actual.

Â¿QuÃ© es React?

React es una biblioteca declarativa, eficiente y flexible para construir interfaces de usuario. Se basa en el concepto de componentes reutilizables que encapsulan su propio estado y lÃ³gica.

Â¿Por quÃ© usar React?

â€¢ Componentes Reutilizables: CÃ³digo modular y mantenible
â€¢ Virtual DOM: Renderizado eficiente y rÃ¡pido
â€¢ Ecosistema Rico: Miles de bibliotecas y herramientas
â€¢ Comunidad Activa: Soporte masivo y recursos abundantes
â€¢ Demanda Laboral: Una de las tecnologÃ-as mÃ¡s solicitadas
â€¢ Flexibilidad: Se puede integrar gradualmente

Historia y EvoluciÃ³n

â€¢ 2011: Jordan Walke crea React en Facebook
â€¢ 2013: React se hace open source
â€¢ 2015: React Native para desarrollo mÃ³vil
â€¢ 2016: React 15 - Mayor estabilidad
â€¢ 2017: React 16 (Fiber) - Nueva arquitectura
â€¢ 2019: React Hooks - RevoluciÃ³n en el manejo de estado
â€¢ 2022: React 18 - Renderizado concurrente

Conceptos Fundamentales

Componentes
Bloques de construcciÃ³n reutilizables que encapsulan UI y lÃ³gica.

JSX (JavaScript XML)
Sintaxis que permite escribir HTML dentro de JavaScript.

Props
Propiedades que se pasan de componentes padre a hijo.

State
Datos internos del componente que pueden cambiar.

Virtual DOM
Representación en memoria del DOM real para optimizar actualizaciones.

4.3 JSX y Componentes BÃ¡sicos

Â¿QuÃ© es JSX?

JSX es una extensiÃ³n de sintaxis para JavaScript que permite escribir elementos HTML dentro del cÃ³digo JavaScript de forma natural.

```
// JavaScript normal
React.createElement('h1', null, 'Â¡Hola Mundo!')

// Con JSX
<h1>Â¡Hola Mundo!</h1>
```

Reglas de JSX

â€¢ Debe devolver un solo elemento padre
â€¢ Los elementos deben cerrarse
â€¢ Usa className en lugar de class
â€¢ Usa camelCase para atributos
â€¢ Usa {} para expresiones JavaScript

```
// Correcto
function MiComponente() {
  return (
    <div>
      <h1 className="titulo">TÃ-tulo</h1>
      <p>PÃ¡rrafo</p>
      <img src="imagen.jpg" alt="DescripciÃ³n" />
    </div>
  );
}
```

Componentes Funcionales

Los componentes son funciones que devuelven JSX:

```
// Componente simple
function Saludo() {
  return <h1>Â¡Hola desde React!</h1>;
}

// Componente con arrow function
const Despedida = () => {
  return <h2>Â¡Hasta la vista!</h2>;
};

// Usar los componentes
function App() {
  return (
    <div>
      <Saludo />
      <Despedida />
    </div>
  );
}
```

Props: Pasando Datos a Componentes

Las props permiten pasar datos de un componente padre a un hijo:

```
// Componente que recibe props
function SaludoPersonalizado(props) {
```

4.4 Estado y Hooks

Â¿QuÃ© es el Estado?

El estado es informaciÃ³n que el componente mantiene internamente y que puede cambiar durante el ciclo de vida del componente. Cuando el estado cambia, React re-renderiza el componente automÃ¡ticamente.

Hook useState

useState es el hook mÃ¡s bÃ¡sico para manejar estado en componentes funcionales:

```
import React, { useState } from 'react';
function Contador() {
  // Declarar estado con valor inicial
  const [contador, setContador] = useState(0);
  const incrementar = () => {
    setContador(contador + 1);
  };
  const decrementar = () => {
    setContador(contador - 1);
  };
  return (
    <div>
      <h2>Contador: {contador}</h2>
      <button onClick={incrementar}>+</button>
      <button onClick={decrementar}>-</button>
    </div>
  );
}
```

Estado con Objetos y Arrays

```
function FormularioUsuario() {
  const [usuario, setUsuario] = useState({
    nombre: '',
    email: '',
    edad: 0
  });
  const actualizarNombre = (nuevoNombre) => {
    setUsuario({
      ...usuario,  // Spread operator para mantener otros valores
      nombre: nuevoNombre
    });
  };
  return (
    <div>
      <input
        type="text"
        value={usuario.nombre}
        onChange={(e) => actualizarNombre(e.target.value)}
        placeholder="Nombre"
      />
      <p>Hola, {usuario.nombre}</p>
    </div>
  );
}
```

4.5 Hook useEffect

Â¿QuÃ© es useEffect?

useEffect permite realizar efectos secundarios en componentes funcionales, como llamadas a APIs, suscripciones, o manipulaciÃ³n del DOM. Es el equivalente a componentDidMount, componentDidUpdate y componentWillUnmount.

Efectos BÃ¡sicos

```
import React, { useState, useEffect } from 'react';
function ContadorConEfecto() {
  const [contador, setContador] = useState(0);
  // Efecto que se ejecuta despuÃ©s de cada render
  useEffect(() => {
    document.title = `Contador: ${contador}`;
  });
  return (
    <div>
      <h2>Contador: {contador}</h2>
      <button onClick={() => setContador(contador + 1)}>+</button>
    </div>
  );
}
```

Efectos con Dependencias

Controlar cuÃ¡ndo se ejecuta el efecto:

```
function UsuarioInfo({ userId }) {
  const [usuario, setUsuario] = useState(null);
  const [cargando, setCargando] = useState(true);
  // Efecto que solo se ejecuta cuando userId cambia
  useEffect(() => {
    const cargarUsuario = async () => {
      setCargando(true);
      try {
        const response = await fetch(`/api/usuarios/${userId}`);
        const userData = await response.json();
        setUsuario(userData);
      } catch (error) {
        console.error('Error cargando usuario:', error);
      } finally {
        setCargando(false);
      }
    };
    cargarUsuario();
  }, [userId]);  // Solo se ejecuta cuando userId cambia
  if (cargando) return <div>Cargando...</div>;
  return (
    <div>
      <h2>{usuario?.nombre}</h2>
      <p>{usuario?.email}</p>
    </div>
  );
}
```

Efectos con Limpieza

4.6 Manejo de Eventos y Formularios

Eventos en React

React usa SyntheticEvents que proporcionan una API consistente:

```
function EventosBasicos() {
  const manejarClick = (e) => {
    console.log('BotÃ³n clickeado!', e);
  };
  const manejarMouseOver = () => {
    console.log('Mouse sobre el elemento');
  };
  const manejarKeyPress = (e) => {
    if (e.key === 'Enter') {
      console.log('Enter presionado');
    }
  };
  return (
    <div>
      <button onClick={manejarClick}>Hacer Click</button>
      <div onMouseOver={manejarMouseOver}>Pasa el mouse aquÃ-</div>
      <input onKeyPress={manejarKeyPress} placeholder="Presiona Enter" />
    </div>
  );
}
```

Formularios Controlados

En React, los formularios controlados mantienen el estado en React:

```
function FormularioContacto() {
  const [datos, setDatos] = useState({
    nombre: '',
    email: '',
    mensaje: '',
    acepta: false
  });
  const manejarCambio = (e) => {
    const { name, value, type, checked } = e.target;
    setDatos({
      ...datos,
      [name]: type === 'checkbox' ? checked : value
    });
  };
  const manejarEnvio = (e) => {
    e.preventDefault();
    console.log('Datos del formulario:', datos);
    // AquÃ- enviarÃ-as los datos al servidor
  };
  return (
    <form onSubmit={manejarEnvio}>
      <input
        type="text"
        name="nombre"
        value={datos.nombre}
        onChange={manejarCambio}
        placeholder="Nombre"
        required
```

4.7 Proyecto Final: AplicaciÃ³n de Tareas Completa

Vamos a crear una aplicaciÃ³n completa que integre todos los conceptos aprendidos:

```
import React, { useState, useEffect } from 'react';
import './App.css';

function App() {
  const [tareas, setTareas] = useState([]);
  const [nuevaTarea, setNuevaTarea] = useState('');
  const [filtro, setFiltro] = useState('todas');
  const [editandoId, setEditandoId] = useState(null);
  const [textoEditando, setTextoEditando] = useState('');

  // Cargar tareas del localStorage al iniciar
  useEffect(() => {
    const tareasGuardadas = localStorage.getItem('tareas');
    if (tareasGuardadas) {
      setTareas(JSON.parse(tareasGuardadas));
    }
  }, []);

  // Guardar tareas en localStorage cuando cambien
  useEffect(() => {
    localStorage.setItem('tareas', JSON.stringify(tareas));
  }, [tareas]);

  const agregarTarea = () => {
    if (nuevaTarea.trim() !== '') {
      const tarea = {
        id: Date.now(),
        texto: nuevaTarea.trim(),
        completada: false,
        fechaCreacion: new Date().toLocaleDateString()
      };
      setTareas([...tareas, tarea]);
      setNuevaTarea('');
    }
  };

  const eliminarTarea = (id) => {
    setTareas(tareas.filter(tarea => tarea.id !== id));
  };

  const toggleCompletada = (id) => {
    setTareas(tareas.map(tarea =>
      tarea.id === id ? { ...tarea, completada: !tarea.completada } : tarea
    ));
  };

  const iniciarEdicion = (id, texto) => {
    setEditandoId(id);
    setTextoEditando(texto);
  };

  const guardarEdicion = () => {
    setTareas(tareas.map(tarea =>
      tarea.id === editandoId ? { ...tarea, texto: textoEditando } : tarea
    ));
    setEditandoId(null);
    setTextoEditando('');
  };

  const cancelarEdicion = () => {
    setEditandoId(null);
    setTextoEditando('');
```

```
      <ul className="lista-tareas">
        {tareasFiltradas.map(tarea => (
          <li key={tarea.id} className={`tarea ${tarea.completada ? 'completada' : ''}`}>
            {editandoId === tarea.id ? (
              <div className="editar-tarea">
                <input
                  type="text"
                  value={textoEditando}
                  onChange={(e) => setTextoEditando(e.target.value)}
                  onKeyPress={(e) => e.key === 'Enter' && guardarEdicion()}
                  onBlur={guardarEdicion}
                  autoFocus
                />
                <button onClick={guardarEdicion}>âœ“</button>
                <button onClick={cancelarEdicion}>âœ—</button>
              </div>
            ) : (
              <div className="ver-tarea">
                <input
                  type="checkbox"
                  checked={tarea.completada}
                  onChange={() => toggleCompletada(tarea.id)}
                />
                <span
                  onDoubleClick={() => iniciarEdicion(tarea.id, tarea.texto)}
                  className="texto-tarea"
                >
                  {tarea.texto}
                </span>
                <small className="fecha">{tarea.fechaCreacion}</small>
                <button
                  onClick={() => eliminarTarea(tarea.id)}
                  className="btn-eliminar"
                >
                  ðŸ—‘ï¸•
                </button>
              </div>
            )}
          </li>
        ))}
      </ul>

      {tareasCompletadas > 0 && (
        <button
          onClick={limpiarCompletadas}
          className="btn-limpiar"
        >
          Limpiar Completadas
        </button>
      )}
    </main>
  </div>
  );
}

export default App;
```

4.8 Resumen del Tema 4

Conceptos Dominados