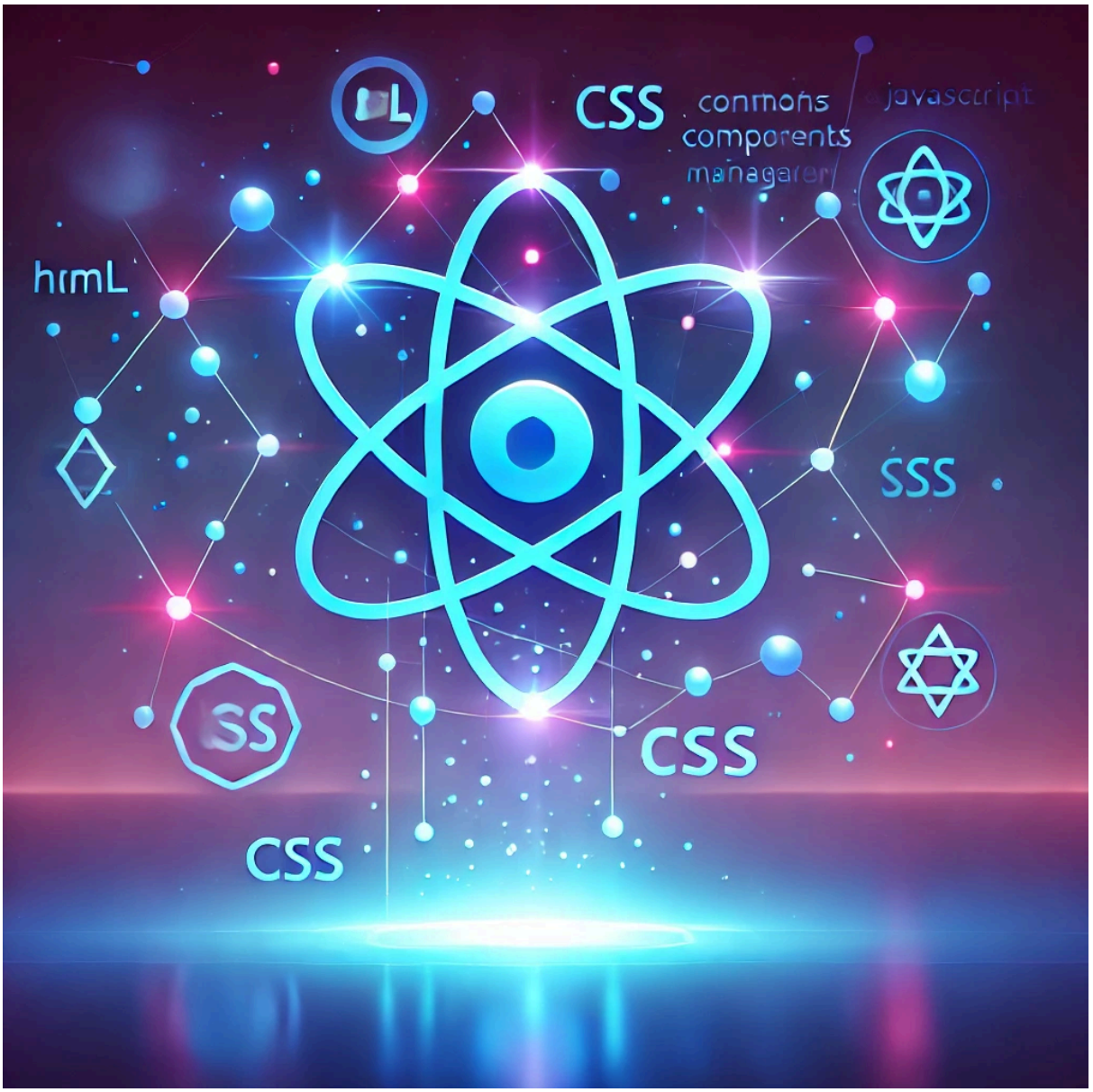
Guía Introductoria a React.js ⚛️



¿Qué es React.js? 🔬

React.js es una biblioteca de JavaScript creada por Facebook que se utiliza para construir interfaces de usuario (UI) de forma eficiente y modular. Se basa en componentes reutilizables que permiten dividir la interfaz en partes pequeñas y manejables.

Beneficios de React.js 🌟

- **Componentes Reutilizables**: Facilita la organización del código al dividir la UI en partes independientes.
- **Virtual DOM**: Acelera las actualizaciones de la interfaz al evitar manipulaciones innecesarias del DOM real.
- **Comunidad Activa**: Amplia documentación, tutoriales y librerías complementarias.
- **Soporte de Hooks**: Permite manejar el estado y el ciclo de vida de forma más eficiente.

Desventajas de React.js 🚫

- **Curva de Aprendizaje**: La combinación de JSX y JavaScript puede ser desafiante para principiantes.
- **Necesidad de Configuraciones**: Proyectos avanzados pueden requerir configuraciones adicionales como Webpack.
- **Actualizaciones Frecuentes**: Los cambios constantes en la biblioteca pueden complicar el mantenimiento.

Instalación con Vite 🚀

1. Asegúrate de tener instalado Node.js.
2. Abre la terminal y ejecuta:

   ```
   npm create vite@latest my-react-app
   ```

3. Si es primera vez que usas vite:

   `y` + enter
4. Selecciona React
5. Selecciona JavaScript
6. Ejecuta:

   ```
   cd my-react-app
   npm install
   ```

7. Si deseas ver tu aplicativo, ejecuta:

   ```
   npm run dev
   ```

8. Abre el navegador en el enlace proporcionado por la terminal (por defecto `http://localhost:5173`).
9. Para cerrar el servidor, presiona en la terminal: ctrl + c

Estructura del Directorio 🌐

- 📄**index.html**: Archivo principal HTML.
- 📁**public**: Archivos estáticos accesibles directamente.
  - 📁**images**: Imágenes del proyecto.
- 📁**src**: Archivos principales de React.
  - 📄**main.jsx**: Punto de entrada del proyecto.
  - 📄**App.jsx**: Componente principal.
  - 📁**components**: Componentes reutilizables.
  - 📁**styles**: Archivos CSS.

Conceptos Básicos 🤖

JSX 🧩

JSX es una extensión de sintaxis que permite escribir HTML dentro de JavaScript. Por ejemplo:

```
const elemento = <h1>Hola, React!</h1>;
```

`<h1>`: Es similar al HTML, pero escrito directamente en JavaScript.

El código JSX debe estar dentro de un solo contenedor. Por ejemplo:

```
return (
  <div>
    <h1>Título</h1>
    <p>Texto</p>
  </div>
);
```

Todo debe estar envuelto en un `<div>` u otro contenedor.

Componentes 📦

Los componentes son bloques reutilizables que definen partes de la UI. Pueden ser funcionales (simples) o de clase (avanzados, menos usados actualmente). Ejemplo de componente funcional:

```
function MiComponente() {
  return <h2>Hola desde un componente</h2>;
}
```

Props 📩

Las props son propiedades que permiten pasar datos entre componentes.

Por notación de punto:

```
function Saludo(props) {
  return <h1>Hola, {props.nombre}!</h1>;
}
```

Uso:

```
<Saludo nombre="Juan" />
```

Por desestructuración:

```
function Saludo({ nombre }) {
  return <h1>Hola, {nombre}!</h1>;
}
```

Estado (State) 🧠

El estado es una variable interna del componente que controla su comportamiento y puede cambiar con el tiempo. Se maneja con `useState` (ver detalles más adelante).

Hooks ⚓

Los hooks son funciones especiales introducidas en React 16.8. Los más comunes son:

- `useState`: Maneja el estado.
- `useEffect`: Maneja efectos secundarios (como llamadas a APIs).

Archivo index.html 📄

Estructura básica del archivo:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>My React App</title>
</head>
<body>
  <div id="root"></div>
  <script type="module" src="/src/main.jsx"></script>
</body>
</html>
```

Archivo main.jsx 📄

Configuración inicial del proyecto:

```
import React from 'react';
import { createRoot } from 'react-dom/client';
import App from './App';

const rootDiv = document.getElementById('root');
const root = createRoot(rootDiv);
root.render(<App />);
```

Explicación:

1. **Importaciones:**
   - `React`: Necesario para usar JSX.
   - `createRoot`: Método para renderizar React en el DOM.
   - `App`: Componente principal.
2. **Constante `rootDiv`:** Obtiene el elemento HTML donde se renderiza la aplicación.
3. **`createRoot`:** Crea un contenedor React para el `div` seleccionado.
4. **Renderizado:** Se renderiza el componente `App`.

Archivo App.jsx 🎨

Estructura de un componente funcional:

```
import React from 'react';

function App() {
  return (
    <div>
      <h1>Hola, React!</h1>
    </div>
  );
}

export default App;
```

Explicación:

1. **Importación de React:** Necesario para usar JSX.
2. **Declaración de la función `App`:** Define el componente principal.
3. **`return`:** Devuelve el JSX que define la UI del componente.
4. **Exportación por defecto:** Permite usar `App` en otros archivos.

Importación de CSS 🎨

Para usar estilos, se pueden importar directamente al archivo:

```
import './styles/App.css';
```

Estilos como Objetos 🖌️

En React, los estilos se pueden definir como objetos:

```
const estiloBoton = {
  backgroundColor: 'blue',
  fontSize: '16px',
  padding: '10px',
};
```

Nota: Las propiedades se escriben en `camelCase` en lugar de `kebab-case`.

- Ejemplo:
  - `background-color` → `backgroundColor`
  - `font-size` → `fontSize`

Notación de Llaves 🪝

Para incluir JavaScript en JSX, se deben usar llaves `{}`:

```
const nombre = 'React';

return <h1>Hola, {nombre}!</h1>;
```

Cambios en Atributos HTML 🏷️

- **`class`** → **`className`**: Para definir clases CSS.
- **`for`** → **`htmlFor`**: Para etiquetas `<label>`.

Directorio `/public` 📁

Para acceder directamente al directorio `/public`, usa `/` en lugar de `./`. Ejemplo:

```
<img src="/images/logo.png" alt="Logo" />
```

Uso de `useState` 🔄

El hook `useState` permite manejar el estado de un componente.

Ejemplo Básico:

```
import React, { useState } from 'react';

function Contador() {
  const [contador, setContador] = useState(0);

  return (
    <div>
      <p>Contador: {contador}</p>
      <button onClick={() => setContador(contador + 1)}>Incrementar</button>
    </div>
  );
}

export default Contador;
```

Explicación:

1. **`useState(0)`:** Define el estado inicial como `0`.
2. **Desestructuración:** Obtiene el estado actual (`contador`) y la función para actualizarlo (`setContador`).
3. **Event Listener:** Al hacer clic en el botón, se llama a `setContador` con el nuevo valor.
4. **Analogía:** Piensa en `useState` como una caja donde guardas un número. `setContador` es la llave para cambiar el valor dentro de la caja. No se puede cambiar el valor de la caja (**const**) sin la llave.

Proyecto 1: Contador de Clicks ➕

Objetivo:

Crear un contador que incremente su valor cada vez que se haga clic en un botón.

Componentes:

- `App.jsx` (Componente principal):

```
import React from 'react';
import Contador from './components/Contador.jsx';

function App() {
  return (
    <div>
      <h1>Contador de Clicks</h1>
      <Contador />
    </div>
  );
}

export default App;
```

- `Contador.jsx` (Lógica del contador):

```
import React, { useState } from 'react';

function Contador() {
  const [contador, setContador] = useState(0);

  return (
    <div>
      <h2>Has hecho clic {contador} veces</h2>
      <button onClick={() => setContador(contador +
1)}>Incrementar</button>
    </div>
  );
}

export default Contador;
```

Proyecto 2: Menú de Restaurante 🍽️

Objetivo:

Crear una lista de cartas que representen platos con nombre, descripción y precio.

Componentes:

- `App.jsx` (Componente principal):

```
import React from 'react';
import Plato from './components/Plato.jsx';

function App() {
  const menu = [
    { nombre: 'Pizza', descripcion: 'Deliciosa pizza con queso',
precio: 10 },
    { nombre: 'Hamburguesa', descripcion: 'Jugosa hamburguesa con
papas', precio: 8 },
    { nombre: 'Ensalada', descripcion: 'Fresca y saludable', precio: 6
},
  ];

  return (
    <div>
      <h1>Menú del Restaurante</h1>
      {menu.map((plato, index) => (
        <Plato
          key={index}
          nombre={plato.nombre}
          descripcion={plato.descripcion}
          precio={plato.precio}
        />
      ))}
    </div>
  );
}

export default App;
```

- `Plato.jsx` (Componente de plato):

```
import React from 'react';

function Plato({ nombre, descripcion, precio }) {
  return (
    <div>
      <h2>{nombre}</h2>
      <p>{descripcion}</p>
      <p>Precio: ${precio}</p>
    </div>
  );
}

export default Plato;
```