Primera Parte: Configuración Inicial del Proyecto "Piedra, Papel, Tijera, Lagarto, Spock"

https://www.youtube.com/watch?v=6YtPXjY30Qc&t=17s&ab_channel=CarlosAzaustre-AprendeJavaScript

Paso 1: Instalación de Node.js

- **Importancia**: Node.js es fundamental para el desarrollo con React, ya que proporciona el entorno de ejecución necesario y el gestor de paquetes (NPM).
- **Proceso de Instalación:**
  - Visita el sitio web oficial de *Node.js.*
  - Descarga e instala la versión recomendada para tu sistema operativo.
  - Una vez instalado, puedes verificar la instalación abriendo tu terminal y ejecutando los comandos **node -v** y **npm -v**. Esto mostrará las versiones de Node.js y NPM, respectivamente.

Paso 2: Crear Proyecto React con Create React Vite

- **¿Por qué Create React Vite?:** Create React Vite es una herramienta moderna para iniciar proyectos de React. Ofrece un inicio más rápido y una mejor experiencia de desarrollo en comparación con **create-react-app**, especialmente para proyectos más grandes.
- **Pasos para Crear el Proyecto:**
  - Abre tu terminal o línea de comandos.
  - Ejecuta el siguiente comando para crear un nuevo proyecto React llamado **react-pptls:**

```
npx create-react-vite react-pptls
```

  - Este comando descargará las dependencias necesarias y configurará la estructura básica del proyecto.
  - Una vez completado el proceso, navega a la carpeta del proyecto ejecutando:

```
cd react-pptls
```

- **Estructura del Proyecto:** Al finalizar, tendrás un nuevo directorio **react-pptls** con la siguiente estructura básica:
  - **node_modules/:** Carpeta donde se almacenan las dependencias del proyecto.
  - **public/**: Carpeta para archivos públicos como el **index.html**.
  - **src/:** Carpeta donde desarrollarás tu juego, incluyendo componentes de React, hojas de estilo y lógica del juego.
  - **package.json:** Archivo que gestiona las dependencias y scripts del proyecto.
  - Otros archivos de configuración y documentación.

Siguiente Paso: Desarrollo del Juego

- Una vez configurado el proyecto, el siguiente paso será desarrollar los componentes, la lógica del juego y la interfaz de usuario, lo que se detallará en las siguientes secciones del tutorial.

Segunda Parte: Estructura del Proyecto "Piedra, Papel, Tijera, Lagarto, Spock"

Paso 1: Organización del Proyecto

La estructura adecuada del proyecto es crucial para mantener tu código organizado y fácilmente mantenible. En este paso, configuraremos la estructura del directorio del proyecto y crearemos los archivos necesarios para el juego.

- **Directorio src/data:**
  - Crea un directorio **data** dentro de **src**.
  - Dentro de **src/data**, crea un archivo llamado **options.js**.
  - Este archivo contendrá las opciones del juego (piedra, papel, tijera, lagarto, spock) junto con sus respectivas reglas.
- **Directorio src/hooks:**
  - Crea un directorio **hooks** dentro de **src.**
  - En **src/hooks**, crea un archivo **useChoices.js**.
  - Este hook personalizado **useChoices** manejará la lógica del juego, incluyendo la selección del jugador, la elección aleatoria del computador y la determinación del ganador.
- **Directorio src/components:**
  - Crea un directorio **components** dentro de **src.**
  - Dentro de **src/components**, crea los siguientes archivos:
    - **OptionButton.jsx**: Componente para mostrar cada opción del juego (piedra, papel, etc.).

- **ResultDisplay.jsx**: Componente para mostrar el resultado de cada ronda.
- **MessageDisplay.jsx**: Componente para mostrar mensajes al jugador.
- **GameControls.jsx:** Componente para controles adicionales del juego, como un botón para reiniciar.

Paso 2: Componente Principal **Game**

- **Archivo src/Game.js:**
  - En **Game.js**, que será el componente principal del juego, integrarás todos los componentes y la lógica del juego.
  - Este componente importará y utilizará **useChoices** para manejar el estado y las acciones del juego.
  - También renderizará los componentes de la interfaz de usuario, como los botones de opción y las pantallas de resultado.

Estructura de Archivos y Directorios

Tu proyecto ahora debería tener la siguiente estructura de archivos y directorios:

```
react-pptls/
├─ node_modules/
├─ public/
├─ src/
│  ├─ components/
│  │  ├─ GameControls.jsx
│  │  ├─ MessageDisplay.jsx
│  │  ├─ OptionButton.jsx
│  │  └─ ResultDisplay.jsx
│  ├─ data/
│  │  └─ options.jsx
│  ├─ hooks/
│  │  └─ useChoices.jsx
│  └─ Game.jsx
├─ package.json
└─ ...otros archivos de configuración...
```

Una vez que la estructura del proyecto esté configurada, el siguiente paso será implementar la lógica del juego en **useChoices.js**, definir las opciones del juego en **options.js**, y desarrollar los componentes de la interfaz de usuario en el directorio **components**.

Tercera Parte: Implementación de Componentes

Componente **Game**

- **Ubicación y Propósito:**
  - **src/Game.jsx** es el corazón de tu juego, donde se reúne toda la lógica y se visualiza la interfaz de usuario.
  - Este componente importa y utiliza el hook **useChoices** para manejar la lógica del juego y renderiza los componentes que forman la interfaz del juego.
- **Desarrollo:**
  - Dentro de **Game.js**, importa los componentes necesarios y el hook **useChoices**.
  - Utiliza el estado proporcionado por **useChoices** para manejar las elecciones del jugador, la lógica del juego y mostrar los resultados.
  - Renderiza los componentes de la interfaz, como botones de opción y mensajes, según el estado actual del juego.

Hook **useChoices**

- **Ubicación y Funcionalidad:**
  - **src/hooks/useChoices.jsx** manejará la lógica detrás de las elecciones del jugador y del computador, y cómo estas elecciones afectan el resultado del juego.
  - El hook utiliza **useState** para controlar el estado del juego y **useEffect** para manejar los efectos secundarios basados en los cambios de estado.
- **Desarrollo:**
  - Define cómo el jugador y el computador hacen sus elecciones.
  - Utiliza **useState** para almacenar las elecciones y los resultados.
  - Implementa la función **getResult** para determinar el ganador de cada ronda.

Datos del Juego (**options.js**)

- **Definición de Opciones:**
  - En **src/data/options.js**, define las posibles elecciones del juego (piedra, papel, tijera, lagarto, spock) y sus interacciones.
  - Cada opción deberá tener un identificador, nombre, emoji representativo y un arreglo de identificadores de las opciones a las que puede ganar.

Componentes de la Interfaz

- **Desarrollo de Componentes Individuales:**
  - En el directorio **src/components/,** crea los componentes que se utilizarán en la interfaz del juego:
    - **OptionButton.js:** Para que el jugador haga su elección.
    - **ResultDisplay.js:** Para mostrar el resultado de la ronda.
    - **MessageDisplay.js**: Para mostrar mensajes informativos o de estado del juego.
    - **GameControls.js**: Para controles adicionales, como un botón para reiniciar el juego.
- **Integración con el Componente Game**:
  - Estos componentes se importarán y utilizarán en **Game.js**.
  - Asegúrate de pasar las **props** necesarias desde **Game** a cada componente para su correcta funcionalidad y visualización.

Conclusión de la Implementación

Con estos pasos, tendrás la base del juego configurada. El componente **Game** servirá como el núcleo del juego, interactuando con el hook **useChoices** para manejar la lógica del juego y utilizando varios componentes para la interfaz. Los datos definidos en **options.jsx** establecerán las reglas y opciones disponibles en el juego.

El siguiente paso será trabajar en la estilización y presentación del juego para crear una experiencia de usuario atractiva y dinámica.

Cuarta Parte: Estilos y Presentación

4. Estilos y Presentación

El aspecto visual es un componente esencial en la creación de juegos, ya que mejora la experiencia del usuario y aporta carácter a tu juego. Vamos a explorar cómo puedes utilizar CSS o Tailwind CSS para estilizar tu juego "Piedra, Papel, Tijera, Lagarto, Spock" en React.

Opción 1: Uso de CSS Puro

- **Creación de Hojas de Estilo:**
  - Crea archivos CSS específicos para cada componente en tu directorio **src/components/.** Por ejemplo, **OptionButton.css**, **ResultDisplay.css**, etc.
  - En cada archivo CSS, define los estilos que corresponden al componente respectivo.
- **Estilización:**
  - Utiliza selectores de clase o ID para aplicar estilos.
  - Define estilos para botones, mensajes, el área de juego, etc., para hacerlos visualmente atractivos.
  - Ejemplo:

```css
.button {
    background-color: blue;
    color: white;
    padding: 10px 20px;
    border-radius: 5px;
    cursor: pointer;
  }
```

- **Integración en Componentes React:**
  - Importa las hojas de estilo en tus componentes de React. Por ejemplo, en **OptionButton.js**, agrega **import './OptionButton.css';.**

Opción 2: Uso de Tailwind CSS

- **Instalación de Tailwind CSS:**
  - Sigue la guía oficial de instalación para integrar Tailwind CSS en tu proyecto React.
  - Esto generalmente implica instalar tailwindcss a través de npm y configurar tu **postcss.config.js.**

- **Aplicación de Estilos con Tailwind CSS:**
  - Utiliza las clases utilitarias de Tailwind CSS directamente en tus componentes JSX para aplicar estilos.
  - Tailwind facilita la creación de diseños responsivos y la aplicación de estilos como padding, margin, colores, y más, directamente en el marcado.

  - Ejemplo:

```
<button className="bg-blue-500 text-white px-4 py-2 rounded
hover:bg-blue-700">
  Jugar
</button>
```

Consejos Generales de Diseño

- **Consistencia Visual:** Mantén una paleta de colores y un estilo consistentes a lo largo de todo el juego.
- **Responsividad:** Asegúrate de que tu juego se vea bien en diferentes dispositivos y tamaños de pantalla.
- **Feedback Visual:** Usa estilos para dar feedback visual a los jugadores, como cambiar el aspecto de los botones al pasar el mouse o al hacer clic.

El diseño y los estilos son fundamentales para hacer que tu juego sea atractivo y agradable de usar. Ya sea mediante CSS puro o utilizando un framework como Tailwind CSS, dedica tiempo a crear una experiencia visual coherente y estéticamente agradable. Esto no solo mejorará la apariencia de tu juego, sino que también puede mejorar la interacción y la experiencia general del usuario.

Quinta Parte: Ejecución y Pruebas

5. Ejecución y Pruebas

Después de desarrollar los componentes y aplicar los estilos, el siguiente paso es ejecutar y probar el juego. Esta fase es crucial para asegurarse de que todas las partes del juego funcionan como se espera y ofrecen una experiencia de usuario fluida y agradable.

Ejecución del Juego

- **Iniciar el Servidor de Desarrollo:**
  - Abre tu terminal o línea de comandos.
  - Navega al directorio de tu proyecto (**react-pptls**).
  - Ejecuta el comando **npm run dev** para iniciar el servidor de desarrollo de Vite.
  - Este comando compilará tu aplicación y la servirá en un servidor local. Normalmente, podrás acceder a tu juego en tu **localhost** con el puerto que se mostrará en la terminal.

Pruebas del Juego

- **Interacción y Funcionalidad**:
  - Prueba cada aspecto del juego:
    - Asegúrate de que cada botón de opción (piedra, papel, tijera, lagarto, spock) funcione correctamente.
    - Comprueba la lógica del juego: ¿se determinan correctamente los ganadores y los perdedores?
    - Verifica que los mensajes y resultados se muestren adecuadamente.
  - Observa la reacción del juego a diferentes tipos de interacciones.
- **Experiencia del Usuario:**
  - Evalúa la interfaz de usuario desde la perspectiva del jugador:
    - ¿Es intuitiva y fácil de entender?
    - ¿Proporciona un feedback visual claro?
    - ¿Es visualmente atractiva y coherente?
- **Rendimiento y Compatibilidad:**
  - Prueba el juego en diferentes navegadores y dispositivos (si es posible) para asegurar la compatibilidad y el rendimiento.
  - Atiende a la velocidad de carga, la fluidez de las animaciones y cualquier posible error o retraso.

Sexta Parte: Creación de Navegación y Routing

6. Navegación y Routing en React

La implementación de un sistema de navegación y routing permite a los usuarios desplazarse entre diferentes páginas o vistas de tu juego, como la página de inicio, el juego en sí y una página de contacto. Para esto, utilizaremos **React Router**, una biblioteca popular para el manejo de rutas en aplicaciones React.

Paso 1: Instalación de React Router

- **Instalación:**
  - En tu terminal, asegúrate de estar en el directorio de tu proyecto y ejecuta:

```
npm install react-router-dom
```

  - Esto instalará **react-router-dom**, que es la versión de React Router para aplicaciones web.

Paso 2: Configuración de React Router

- **Modificación de App.js:**
  - En tu archivo **App.js**, importa **BrowserRouter**, **Routes** y **Route** de **react-router-dom**.
  - Define las rutas para las páginas del juego. Por ejemplo:

```
import { BrowserRouter as Router, Routes, Route } from 'react-
router-dom';
import Home from './components/Home';
import Game from './components/Game';
import Contact from './components/Contact';

function App() {
  return (
    <Router>
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/game" element={<Game />} />
        <Route path="/contact" element={<Contact />} />
      </Routes>
    </Router>
  );
}
```

- Aquí, **Home**, **Game** y **Contact** son componentes que representan las diferentes páginas de tu juego.

Paso 3: Creación de Componentes de Página

- **Componentes de Página:**
  - Crea componentes React para cada página. Por ejemplo, **Home.jsx**, **Game.jsx** y **Contact.jsx** en el directorio **src/components/.**
  - En **Home.jsx**, incluye una descripción del juego y un botón o enlace para comenzar el juego.
  - **Game.jsx** contendrá la lógica y la visualización de tu juego.
  - **Contact.jsx** puede incluir información de contacto o un formulario para que los usuarios se comuniquen contigo.

Paso 4: Creación de un Componente de Navegación

- **Componente de Navegación:**
  - Crea un componente **Navbar.jsx** para la navegación.
  - Utiliza el componente **Link** de **react-router-dom** para crear enlaces que permitan a los usuarios navegar entre las páginas.
  - Ejemplo:

```
import { Link } from 'react-router-dom';

function Navbar() {
  return (
    <nav>
      <Link to="/">Home</Link>
      <Link to="/game">Game</Link>
      <Link to="/contact">Contact</Link>
    </nav>
  );
}
```

  - Importa y utiliza **Navbar** en tu componente **App.js.**

Paso 5: Estilización del Navegador

- **Estilos para el Navegador:**
  - Utiliza CSS o Tailwind CSS para dar estilos al navegador, asegurando que se integre bien con el diseño general de tu juego.
  - Considera estilos para los enlaces activos o al pasar el mouse.

La implementación de navegación y routing en tu juego React facilita la movilidad entre diferentes secciones, como la página de inicio, el juego y la página de contacto, creando una experiencia de usuario más completa y profesional. Con **React Router**, puedes configurar fácilmente estas rutas y ofrecer una navegación fluida y coherente en tu juego.

Nueva Estructura del Proyecto con Navegación y Routing

Con la adición de navegación y routing, la estructura de tu proyecto "Piedra, Papel, Tijera, Lagarto, Spock" en React se ha ampliado para incluir nuevos componentes y la configuración de rutas. Aquí está la estructura actualizada:

```
react-pptls/
├─ node_modules/
├─ public/
│  ├─ index.html
│  └─ ...otros archivos...
├─ src/
│  ├─ components/
│  │  ├─ Game.jsx
│  │  ├─ GameControls.jsx
│  │  ├─ MessageDisplay.jsx
│  │  ├─ OptionButton.jsx
│  │  ├─ ResultDisplay.jsx
│  │  ├─ Navbar.jsx        # Nuevo componente para la navegación
│  │  ├─ Home.jsx          # Nuevo componente para la página de inicio
│  │  ├─ Contact.jsx       # Nuevo componente para la página de contacto
│  │  └─ ...otros componentes...
│  ├─ data/
│  │  └─ options.js
│  ├─ hooks/
│  │  └─ useChoices.jsx
│  ├─ App.jsx              # Modificado para incluir routing
│  ├─ main.jsx            # Punto de entrada del proyecto
│  └─ ...otros archivos...
├─ package.json
├─ vite.config.js         # Configuración de Vite (si estás usando Vite)
└─ ...otros archivos de configuración...
```

Descripción de los Cambios

- **src/components/:**
  - Se han añadido nuevos componentes para manejar diferentes páginas y la navegación:
    - **Navbar.js:** Componente para la barra de navegación que permite moverse entre diferentes páginas.
    - **Home.js:** Componente para la página de inicio, donde se presenta el juego y se ofrece un enlace o botón para empezar a jugar.
    - **Contact.js:** Componente para la página de contacto, donde los jugadores pueden encontrar información de contacto o enviar mensajes.
- **src/App.js:**
  - Modificado para incluir la lógica de routing con **React Router**.
  - Importa y utiliza **Routes** y **Route** de **react-router-dom** para definir las rutas de las diferentes páginas.
  - Integra el componente **Navbar** para la navegación entre páginas.

Notas Adicionales

- **Estilos:**
  - Asegúrate de que cada nuevo componente tenga su correspondiente hoja de estilos si estás usando CSS puro, o define las clases de Tailwind CSS directamente en los componentes si optas por Tailwind.
- **Pruebas:**
  - Después de hacer estos cambios, ejecuta y prueba tu aplicación para asegurarte de que la navegación entre páginas funciona correctamente y que todos los componentes se renderizan como se espera.