**Ejercicio 1: Primeros Pasos y Flujo de Trabajo Básico 🚀**

El objetivo es que los alumnos se familiaricen con el ciclo de trabajo fundamental de Git: **clonar, modificar, confirmar y subir cambios**.

1. **Creación de un Repositorio (Repo):**
   * Pide a cada alumno que cree una cuenta en GitHub si no la tienen.
   * Indícales que creen un **nuevo repositorio público** llamado mi-primer-ejercicio.
   * Deben inicializarlo con un archivo **README.md**.
2. **Clonación Local:**
   * Cada alumno debe clonar este repositorio a su computadora usando el comando: git clone <URL\_del\_repo>.
   * Explícales la importancia de la URL del repositorio.
3. **Modificación y Confirmación (Commit):**
   * En su repositorio local, deben crear un archivo simple, como index.html o saludo.txt, y escribir un mensaje de bienvenida.
   * Pídeles que agreguen el archivo al área de **staging** con git add . o git add index.html.
   * Luego, deben confirmar el cambio con un mensaje descriptivo: git commit -m "Añadir archivo de saludo".
4. **Subir Cambios (Push):**
   * Finalmente, deben subir sus cambios al repositorio remoto en GitHub usando: git push origin main.
   * Puedes explicarles que origin es el nombre predeterminado para el repositorio remoto y main es la rama principal.

**Ejercicio 2: Colaboración y Ramas (Branches) 🤝**

Este ejercicio se centra en el trabajo en equipo y el uso de ramas para evitar conflictos y gestionar el desarrollo de nuevas funcionalidades.

1. **Crear una Rama:**
   * Forma equipos de 2 o 3 alumnos.
   * Uno de los alumnos debe crear una nueva rama desde la rama principal (main) con el comando: git branch nueva-funcionalidad.
   * Luego, debe cambiarse a esa rama con: git checkout nueva-funcionalidad.
   * También puedes enseñarles a hacerlo en un solo paso con: git checkout -b nueva-funcionalidad.
2. **Trabajar en la Rama:**
   * Dentro de la nueva rama, cada alumno del equipo debe modificar un archivo o añadir uno nuevo. Por ejemplo, pueden crear un archivo llamado equipo.md y cada uno debe agregar su nombre y una frase.
   * Cada alumno debe seguir el flujo de trabajo básico: git add, git commit, y git push origin nueva-funcionalidad.
   * Asegúrate de que cada miembro del equipo suba sus cambios a la **misma rama**.
3. **Fusionar Cambios (Merge) y Pull Request:**
   * Uno de los alumnos del equipo debe ir a la interfaz de GitHub.
   * Ahí, verán la opción de **crear un Pull Request** (PR) para fusionar su rama nueva-funcionalidad con la rama principal main.
   * Pídeles que describan los cambios en el PR.
   * El otro miembro del equipo (o tú como instructor) puede revisar el PR, dejar comentarios y finalmente **fusionar** la rama.

**Ejercicio 3: Resolución de Conflictos y GitHub Pages 💥**

Este ejercicio enseña a los alumnos a manejar un problema común en la colaboración: los conflictos de fusión. También introduce la publicación de un sitio web.

1. **Crear un Conflicto Intencional:**
   * Pídeles que trabajen en un mismo archivo, por ejemplo about.html, desde la rama main.
   * Dos alumnos deben modificar la misma línea del archivo al mismo tiempo (por ejemplo, el título <h1>).
   * El primer alumno sube sus cambios con git push origin main.
   * El segundo alumno intenta subir sus cambios. GitHub rechazará el push y le pedirá que primero actualice su repositorio local con git pull.
   * Cuando ejecute git pull, se generará un **conflicto de fusión**.
2. **Resolver el Conflicto:**
   * Explica cómo los marcadores de conflicto (<<<<<<<, =======, >>>>>>>) aparecen en el archivo.
   * Guíalos para que elijan la versión correcta del código, eliminen los marcadores de conflicto, guarden el archivo y sigan el proceso de git add, git commit y git push para resolverlo.
3. **Publicar con GitHub Pages:**
   * Una vez resuelto el conflicto, pídeles que publiquen su proyecto.
   * En la configuración del repositorio en GitHub, deben ir a la sección de **GitHub Pages**.
   * Pídeles que seleccionen la rama main y la carpeta raíz (/) como fuente de publicación.
   * Después de unos minutos, su sitio web estará accesible en una URL como <nombre\_usuario>.github.io/<nombre\_del\_repo>.

**----------------------------------------------------------------------------------------------------------------------------------------------------------**

**1. Ejercicio: Guarda tu Nombre 🧑‍💻**

Este ejercicio es el más simple de todos y se centra en la operación de guardar y recuperar un solo dato.

**Instrucciones:**

1. En el HTML, crea un **input de texto** y un **botón** que diga "Guardar".
2. En el JavaScript, añade un **evento de clic** al botón.
3. Cuando el botón sea clicado, toma el valor del input y guárdalo directamente en localStorage bajo una clave como "nombreUsuario". No necesitas usar JSON aquí, ya que es un simple string.
4. Al cargar la página, revisa si la clave "nombreUsuario" existe en localStorage. Si es así, toma ese valor y muéstralo en la página, por ejemplo, en un párrafo con un mensaje como: "¡Hola de nuevo, [nombre guardado]!".
5. Si no hay un nombre guardado, no muestres nada o muestra un mensaje por defecto.

**2. Ejercicio: Contador de Visitas 🔢**

Este ejercicio introduce la idea de leer, modificar y volver a guardar un dato numérico, lo que requiere usar JSON.parse y JSON.stringify.

**Instrucciones:**

1. Crea un párrafo en el HTML que muestre el número de visitas.
2. Al cargar la página en JavaScript:
   * Intenta obtener el valor de localStorage con una clave como "contadorVisitas".
   * **Muy importante:** Los valores de localStorage son siempre strings. Debes usar JSON.parse() para convertirlo a un número. Si el valor es null (la primera vez que se visita la página), inicializa el contador en 1.
   * Si el valor existe, **incrementa el número en 1**.
3. Después de actualizar el número, usa JSON.stringify() para convertirlo de nuevo a un string y **guárdalo en localStorage**.
4. Finalmente, actualiza el texto del párrafo en la página para mostrar el nuevo número de visitas.

**3. Ejercicio: Mis Colores Favoritos 🎨**

Este ejercicio es ideal para que practiquen el manejo de un array simple de strings.

**Instrucciones:**

1. Crea un input de texto para que el usuario escriba un color y un botón "Agregar Color".
2. Crea una lista vacía (<ul>) donde se mostrarán los colores.
3. Cuando se haga clic en "Agregar Color":
   * Lee el valor del input.
   * Intenta obtener un array de colores de localStorage con la clave "coloresFavoritos". Usa JSON.parse() para convertir el string a un array. Si no existe, crea un array vacío [].
   * Agrega el nuevo color al array.
   * Convierte el array actualizado a JSON con JSON.stringify() y guárdalo de nuevo en localStorage.
   * Actualiza la lista en la página para mostrar todos los colores del array.
4. Al cargar la página, si hay colores guardados, cárgalos de localStorage y muéstralos en la lista.

**Más avanzados**

**1. Ejercicio Básico: Lista de Tareas Simple (To-Do List) 📝**

Este ejercicio es ideal para que se familiaricen con los fundamentos.

**Objetivos:**

* **Guardar datos:** Almacenar una lista de tareas en localStorage.
* **Recuperar datos:** Cargar las tareas guardadas al iniciar la página.
* **Manejar JSON:** Convertir un array de tareas a JSON (JSON.stringify()) para guardarlo y de vuelta a un array (JSON.parse()) al recuperarlo.

**Instrucciones:**

1. Crea un input de texto y un botón "Agregar Tarea".
2. Crea una lista (<ul> o <div>) donde se mostrarán las tareas.
3. Cuando el usuario escriba una tarea y haga clic en el botón, el texto se debe agregar a un array en JavaScript.
4. Luego, el array se debe convertir a JSON y guardarse en localStorage bajo una clave como "tareas".
5. Actualiza la lista en la página para mostrar la nueva tarea.
6. Al cargar la página, revisa si hay datos en localStorage con la clave "tareas". Si existen, cárgalos, conviértelos a un array y muestra las tareas en la lista.

**2. Ejercicio Intermedio: Perfil de Usuario 👤**

Este ejercicio profundiza en el manejo de objetos y la persistencia de datos más estructurados.

**Objetivos:**

* **Almacenar objetos:** Guardar un objeto JavaScript en localStorage.
* **Validación de datos:** Comprobar si un objeto existe antes de intentar acceder a sus propiedades.
* **Actualización de datos:** Modificar valores existentes en el objeto y guardarlos nuevamente.

**Instrucciones:**

1. Crea un formulario con varios campos para un perfil de usuario, por ejemplo:
   * Nombre (input de texto)
   * Email (input de email)
   * Edad (input de número)
2. Agrega un botón "Guardar Perfil".
3. Al hacer clic en el botón, recopila los datos del formulario y crea un objeto JavaScript, por ejemplo:

JavaScript

const perfil = { nombre: "Juan", email: "juan@example.com", edad: 30 };

1. Convierte este objeto a JSON y guárdalo en localStorage con una clave "perfilUsuario".
2. Al cargar la página, si "perfilUsuario" existe en localStorage, carga el objeto, conviértelo de JSON a JavaScript y rellena los campos del formulario con los datos guardados.
3. Añade un botón "Limpiar Perfil" que borre los datos de localStorage y vacíe los campos del formulario.

**3. Ejercicio Avanzado: Carrito de Compras 🛒**

Este ejercicio combina los conceptos anteriores en un escenario más realista, ideal para un proyecto final.

**Objetivos:**

* **Manejar arrays de objetos:** Almacenar una lista de productos, donde cada producto es un objeto.
* **Operaciones CRUD:** Agregar, eliminar y actualizar elementos dentro del array guardado.
* **Cálculos dinámicos:** Sumar el total de precios de los productos.

**Instrucciones:**

1. **Crea una lista de productos fijos:** Define un array de objetos en tu código con al menos 3 o 4 productos. Cada objeto debe tener propiedades como id, nombre, precio e imagen (puedes usar URLs de imágenes de prueba).
2. **Muestra los productos:** Renderiza estos productos en la página con un botón "Agregar al Carrito" para cada uno.
3. **Maneja el carrito:**
   * Al hacer clic en "Agregar al Carrito", crea o actualiza un array llamado "carrito" en localStorage. Este array almacenará los productos que el usuario ha seleccionado.
   * **Importante:** Cada vez que se agrega un producto, revisa si ya existe en el carrito. Si existe, aumenta su cantidad en lugar de agregarlo de nuevo.
   * Calcula y muestra el precio total del carrito en la página, actualizándolo cada vez que se agrega o elimina un producto.
4. **Crea un botón "Vaciar Carrito"** que borre el array de productos del localStorage y restablezca la visualización del carrito.