Paso 1: Preparar el Proyecto

1. **Crea la Estructura Básica del Proyecto:**
    - Abre tu editor de texto (recomendado: VS Code).
    - Crea una carpeta llamada HospitalProyecto.
    - Dentro de esta carpeta, crea los siguientes archivos y carpetas:
        - index.html: Será la página principal del proyecto donde se mostrará la interfaz web.
        - style.css: Archivo para agregar estilos al sitio web.
        - script.js: Archivo para escribir toda la lógica en JavaScript.
        - README.md: Documento donde se explicará todo el desarrollo del proyecto.
2. **Instala un Servidor Local (Opcional):**
    - Si quieres probar el proyecto como si estuviera en un servidor real, utilizar el servidor visto en clases.
3. **Abre tu Proyecto:**
    - En el navegador, abre el archivo index.html para ver cómo evoluciona el proyecto mientras trabajas en él.

Paso 2: Crear y Manipular JSON

1. **Define un Objeto JSON para Doctores:**
    - En script.js, empieza definiendo un objeto JSON que contenga información sobre los doctores:

Javascript

```
const doctores = [
 {
  nombre: "Juan Pérez",
  especialidad: "Cardiología",
  experiencia: 10,
  disponibilidad: true,
  contacto: {
   telefono: "123456789",
   email: "juan.perez@hospital.com",
  },
  horarios: ["Lunes 9-12", "Miércoles 10-14"],
 },
 {
  nombre: "Ana López",
  especialidad: "Pediatría",
  experiencia: 5,
  disponibilidad: false,
  contacto: {
   telefono: "987654321",
   email: "ana.lopez@hospital.com",
  },
  horarios: ["Martes 8-13", "Jueves 14-18"],
 },
];
```

2. **Explicación:**
    - El objeto doctores es un arreglo de objetos JSON, donde cada objeto representa un doctor con su nombre, especialidad, experiencia, disponibilidad, contacto y horarios.

3. **Accede a los Datos con Destructuring:**
    - Para acceder a los datos de un doctor, usa la técnica de destructuring:

Javascript

```
const { nombre, especialidad, contacto: { email } } = doctores[0];
console.log(`Doctor: ${nombre}, Especialidad: ${especialidad}, Email: ${email}`);
```

4. **Explicación:**
    - Aquí extraemos directamente las propiedades nombre, especialidad y email del primer doctor, reduciendo el código necesario para acceder a sus valores.
5. **Muestra Información en la Interfaz:**
    - En index.html, agrega un elemento para mostrar los datos:

html

```
<div id="doctores-info"></div>
```

    - En script.js, usa innerHTML para insertar la información del primer doctor:

Javascript

```
const divInfo = document.getElementById("doctores-info");
divInfo.innerHTML = `<h2>${nombre}</h2><p>Especialidad:
${especialidad}</p><p>Email: ${email}</p>`;
```

Paso 3: Operaciones con JSON

1. **Clonación de un Objeto:**
    - Crea una copia de un doctor y modifícalo sin alterar el original:

Javascript

```
const doctorClonado = { ...doctores[0] };
doctorClonado.nombre = "Carlos Gómez";
console.log("Original:", doctores[0]);
console.log("Clonado:", doctorClonado);
```

2. **Explicación:**
    - Se usa el operador de propagación ... para copiar el contenido del objeto doctores[0] a un nuevo objeto llamado doctorClonado.
3. **Fusión de Objetos:**
    - Fusiona información adicional en un nuevo objeto:

Javascript

```
const servicios = { cardiología: true, pediatría: true };
const infoCompleta = { ...doctores[0], ...servicios };
console.log(infoCompleta);
```

4. **Explicación:**
    - Combina las propiedades del objeto doctores[0] con las propiedades del objeto servicios.

5. **Recorrer y Convertir a String JSON:**
    - Muestra la información de todos los doctores en el navegador:

```
javascript

doctores.forEach(doc => {
  console.log(JSON.stringify(doc));
});
```

6. **Explicación:**
    - La función JSON.stringify() convierte un objeto JSON a una cadena legible para ser mostrada en consola o almacenada.

Paso 4: Implementación de Estructuras de Datos

1. **Gestión de Arreglos:**
    - Agrega, elimina y busca doctores:

```
javascript

doctores.push({ nombre: "Nuevo Doctor", especialidad: "Dermatología", experiencia:
3 });
doctores.pop(); // Elimina el último elemento
const encontrado = doctores.find(doc => doc.nombre === "Ana López");
console.log(encontrado);
```

2. **Simular una Pila para Citas:**
    - Implementa una pila para gestionar las citas:

```
Javascript

const citas = [];
citas.push("Cita 1");
citas.push("Cita 2");
console.log(citas.pop()); // Atiende la última cita
```

3. **Simular una Cola para Pacientes:**
    - Implementa una cola para manejar el orden de llegada:

javascript

```
const pacientes = [];
pacientes.push("Paciente 1");
pacientes.push("Paciente 2");
console.log(pacientes.shift()); // Atiende al primer paciente
```

Paso 5: Implementación de Algoritmos

1. **Búsqueda de Doctores:**
    - Implementa una función para buscar doctores por nombre:

javascript

```
const buscarDoctor = nombre => doctores.find(doc => doc.nombre === nombre);
console.log(buscarDoctor("Juan Pérez"));
```

2. **Ordenamiento por Experiencia:**
    - Ordena los doctores por años de experiencia:

javascript

```
doctores.sort((a, b) => b.experiencia - a.experiencia);
console.log(doctores);
```

3. **Explicación:**
    - La función sort ordena los elementos del arreglo de mayor a menor experiencia.

4. **Documentar Complejidad:**
    - En README.md, explica:
        - Búsqueda: O(n) porque revisa cada elemento.
        - Ordenamiento: O(n log n) por el algoritmo sort.

Paso 6: Documentar y Entregar

1. **Documentación:**
    - En README.md, explica:
        - Cómo se crearon y manipularon los objetos JSON.
        - Qué estructuras de datos se usaron y por qué.
        - Cómo funcionan los algoritmos y su complejidad.
2. **Entrega:**
    - Comprime los archivos en un ZIP o súbelos a un repositorio en GitHub.