🎨 10 EJERCICIOS BÁSICOS DE DOM CON DATOS LOCALES

🚀 CONFIGURACIÓN INICIAL (IMPORTANTE)

1️⃣ Crea la carpeta de tu proyecto

```
mkdir ejercicios-dom
cd ejercicios-dom
```

2️⃣ Estructura de carpetas recomendada

```
ejercicios-dom/
├── data.js
├── index.html
├── src/
│   ├── ejercicio1.js
│   ├── ejercicio2.js
│   ├── ejercicio3.js
│   ├── ejercicio4.js
│   ├── ejercicio5.js
│   ├── ejercicio6.js
│   ├── ejercicio7.js
│   ├── ejercicio8.js
│   ├── ejercicio9.js
│   └── ejercicio10.js
└── style.css
```

3️⃣ Importa los datos

En cada archivo JavaScript, deberás importar los datos necesarios desde data.js:

```
import {
  bienvenida, tareas, peliculas, alojamientos, ubicaciones,
  usuariosConHabilidades, publicaciones, menuRestaurante,
  eventos, proyectos
} from '../data.js';
```

🏷 EJERCICIO 1: Obtener un Párrafo Dinámico desde Datos

Nivel: ⭐ Muy Básico

Enunciado

Utiliza los datos del objeto `bienvenida` para crear un párrafo en el DOM con:

- Texto obtenido del objeto `bienvenida.texto`
- Clase CSS OBLIGATORIA: "welcome-message"
- El color azul ya está definido en el CSS para esta clase

Archivo: `src/ejercicio1.js`

Conceptos que practicarás

- ✅ Importación de datos desde otro módulo
- ✅ Acceso a propiedades de objetos
- ✅ document.createElement()
- ✅ appendChild()
- ✅ classList.add()

Resultado Esperado

```
┌──────────────────────────────────────┐
│ Aplicación                           │
├──────────────────────────────────────┤
│                                      │
│  Hola, bienvenido a JavaScript       │ ← Azul
│                                      │
└──────────────────────────────────────┘
```

🏷️ EJERCICIO 2: Crear Lista de Tareas desde Datos

Nivel: ⭐⭐ Fácil

Enunciado

Utiliza el array `tareas` para renderizar cada una en una lista no ordenada (`<ul>`).

- Cada elemento de la lista debe tener la clase CSS "task-item"
- Si la tarea está completada, añadir también la clase "completed"

Archivo: `src/ejercicio2.js`

Conceptos que practicarás

- ✅ Importación de datos

- ✅ Array.forEach()
- ✅ Iterar sobre arrays
- ✅ Crear múltiples elementos
- ✅ appendChild()

Estructura de datos

```
// El array tareas contiene:
[
  { "id": 1, "texto": "Estudiar JavaScript", "completada": false },
  { "id": 2, "texto": "Hacer ejercicios", "completada": false },
  { "id": 3, "texto": "Crear un proyecto", "completada": false }
]
```

Resultado Esperado

```
┌──────────────────────────────────┐
│ Mi Lista de Tareas               │
├──────────────────────────────────┤
│                                  │
│ ✗ Estudiar JavaScript            │
│ ✗ Hacer ejercicios               │
│ ✗ Crear un proyecto              │
│ ✓ Practicar DOM                  │
│ ✗ Leer documentación             │
│                                  │
└──────────────────────────────────┘
```

💡 Pistas

```
// 1. Importar tareas desde data.js
import { tareas } from '../data.js';

// 2. Obtener el elemento app
// 3. Crear encabezado <h2>
// 4. Crear elemento <ul>
// 5. Usar forEach para iterar tareas
// 6. Crear <li> para cada tarea
// 7. Agregar clase "completada" si tarea.completada === true
// 8. Usar operador ternario: tarea.completada ? '✓' : '✗'
```

🏷️ EJERCICIO 3: Tarjetas de Películas desde Datos

Nivel: ⭐⭐⭐ Medio

Enunciado

Utiliza el array `peliculas` para renderizar cada una como una tarjeta (card) con:

- Título (clase CSS: "movie-title")
- Año (clase CSS: "movie-year")
- Rating (clase CSS: "movie-rating", mostrar estrellas ⭐)
- Cada tarjeta debe tener la clase CSS "movie-card"
- El contenedor de todas las tarjetas debe tener la clase "movies-container"

Archivo: `src/ejercicio3.js`

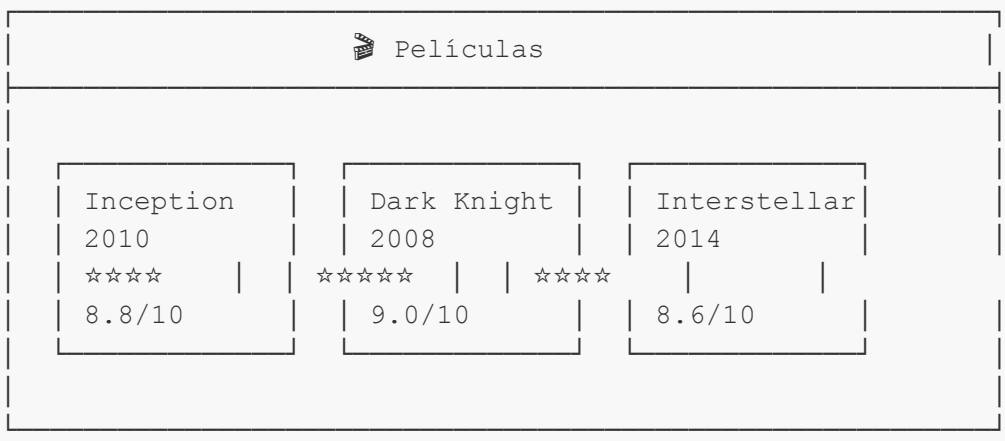
Conceptos que practicarás

- ✅ Importación de datos
- ✅ Objetos y propiedades
- ✅ Iterar arrays de objetos
- ✅ CSS Grid
- ✅ Classes dinámicas
- ✅ Math.floor()

Estructura de datos

```
// El array peliculas contiene:
[
  { "id": 1, "titulo": "Inception", "año": 2010, "rating": 8.8 },
  { "id": 2, "titulo": "The Dark Knight", "año": 2008, "rating": 9.0 },
  { "id": 3, "titulo": "Interstellar", "año": 2014, "rating": 8.6 }
]
```

Resultado Esperado



💡 Pistas

```
// 1. Importar peliculas desde data.js
import { peliculas } from '../data.js';

// 2. Obtener el elemento app
// 3. Crear encabezado <h1>
// 4. Crear contenedor <div class="movies-container">
// 5. Usar forEach para iterar películas
// 6. Para cada película:
//    - Crear <div class="movie-card">
//    - Crear div título
//    - Crear div año
//    - Crear div rating
// 7. Calcular número de estrellas: Math.floor(rating / 2)
// 8. Repetir estrella: '⭐'.repeat(numEstrellas)
```

🏷️ EJERCICIO 4: Tabla de Alojamientos desde Datos

Nivel: ⭐⭐⭐ Medio-Alto

Enunciado

Utiliza el array `alojamientos` para renderizar una tabla HTML con:

- Encabezados: Nombre | Ubicación | Precio | Rating
- Filas dinámicas obtenidas de los datos
- La tabla debe estar dentro de un div con clase "table-container"
- Los precios deben tener la clase CSS "price"
- Los ratings deben tener la clase CSS "rating"

Archivo: `src/ejercicio4.js`

Conceptos que practicarás

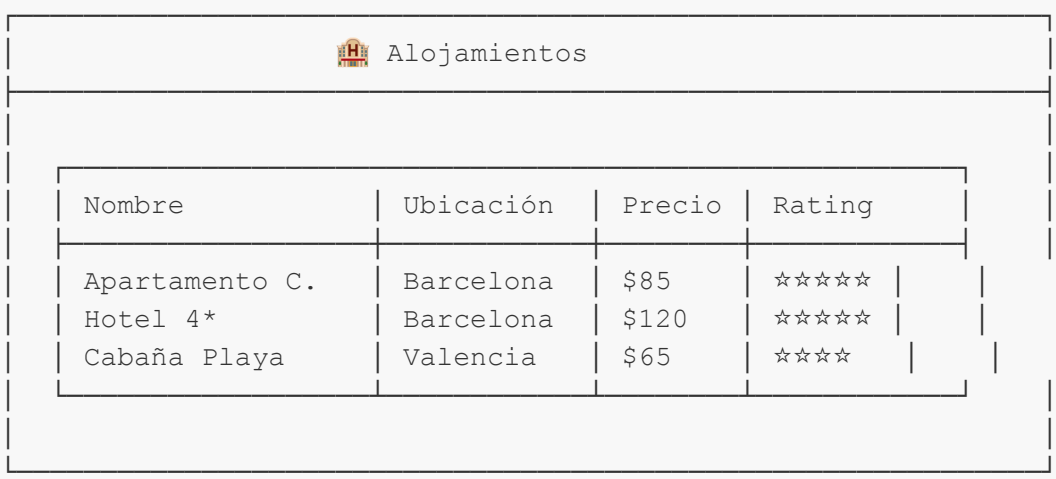
- ✅ Importación de datos
- ✅ Crear tablas dinámicamente
- ✅ `<table>`, `<thead>`, `<tbody>`, `<tr>`, `<td>`
- ✅ Iterar y renderizar datos
- ✅ Formatear datos (precio con $, rating con estrellas)

Estructura de datos

```
// El array alojamientos contiene:
[
```

```
  { "id": 1, "nombre": "Apartamento Centro", "ubicacion": "Barcelona",
"precio": 85, "rating": 4.7 },
  { "id": 2, "nombre": "Hotel 4 Estrellas", "ubicacion": "Barcelona",
"precio": 120, "rating": 4.9 },
  { "id": 3, "nombre": "Cabaña Playa", "ubicacion": "Valencia", "precio":
65, "rating": 4.8 }
]
```

Resultado Esperado



Pistas

```
// 1. Importar alojamientos desde data.js
import { alojamientos } from '../data.js';

// 2. Obtener el elemento app
// 3. Crear encabezado <h1>
// 4. Crear elemento <table>
// 5. Crear <thead> con headers: Nombre, Ubicación, Precio, Rating
// 6. Crear <tbody>
// 7. Usar forEach para iterar alojamientos
// 8. Para cada alojamiento crear <tr>
// 9. Agregar 4 <td>: nombre, ubicación, precio con $, rating con estrellas
// 10. Math.floor(rating) para obtener número de estrellas
```

EJERCICIO 5: Formulario Dinámico con Validación

Nivel: ⭐⭐⭐⭐ Avanzado

Enunciado

Crea un formulario de búsqueda dinámicamente que:

1. Obtiene opciones de ubicación desde el array `ubicaciones`
2. Crea inputs: ubicación (select), check-in (date), check-out (date)
3. Valida que todos los campos estén completos
4. Valida que check-out sea posterior a check-in
5. Muestra mensaje de éxito o error

Clases CSS OBLIGATORIAS:

- Contenedor de mensajes: "message-container"
- Grupo de formulario: "form-group"
- Fila del formulario: "form-row"
- Mensajes de error: "error"
- Mensajes de éxito: "success"

Archivo: `src/ejercicio5.js`

Conceptos que practicarás

- ✅ Importación de datos
- ✅ event.preventDefault()
- ✅ Validación de formularios
- ✅ Comparación de fechas
- ✅ Crear elementos `<select>` dinámicamente
- ✅ Manejo de eventos submit

Estructura de datos

```
// El array ubicaciones contiene:
[
  { "id": 1, "nombre": "Barcelona", "pais": "España" },
  { "id": 2, "nombre": "Valencia", "pais": "España" },
  { "id": 3, "nombre": "Madrid", "pais": "España" }
]
```

Resultado Esperado

```
┌──────────────────────────────────────────────────────┐
│           🏨 Buscar Alojamiento                      │
├──────────────────────────────────────────────────────┤
│                                                      │
│ Ubicación:                                           │
│ [▼ Selecciona ubicación]                             │
│   - Barcelona                                        │
│   - Valencia                                         │
```

```
┌──────────────────────────────────────────────────────────┐
│   - Madrid                                               │
│                                                          │
│ Check-in:           Check-out:                           │
│ [_______________]   [_______________]                    │
│                                                          │
│              [🔍 Buscar]                                 │
│                                                          │
│ ✖ Por favor, completa todos los campos                   │
│  (o)                                                     │
│ ✔ ¡Búsqueda realizada!                                   │
│                                                          │
└──────────────────────────────────────────────────────────┘
```

💡 Pistas

```
// 1. Importar ubicaciones desde data.js
import { ubicaciones } from '../data.js';

// 2. Obtener el elemento app
// 3. Crear encabezado <h1>
// 4. Crear <div> para mensajes (vacío al inicio)
// 5. Crear <form>
// 6. Agregar event listener 'submit' al form
// 7. En el handler:
//    - Llamar event.preventDefault()
//    - Obtener valores de inputs
//    - Validar que no estén vacíos
//    - Validar que checkOut > checkIn (usar new Date())
//    - Mostrar error o éxito en messageContainer
// 8. Crear <select> dinámicamente y llenar con datos
// 9. Usar forEach para agregar <option> por cada ubicación
```

🏷️ EJERCICIO 6: Tarjetas de Usuarios con Habilidades

Nivel: ⭐⭐ Fácil-Medio

Enunciado

Utiliza el array `usuariosConHabilidades` para crear tarjetas de usuario que muestren:
Clases CSS OBLIGATORIAS:

- Contenedor principal: "users-container"
- Tarjeta de cada usuario: "user-card"
- Nombre del usuario: "user-name"
- Edad y email: "user-info"

- Contenedor de habilidades: "skills-container"
- Cada habilidad: "skill-tag"
- Nivel del usuario: "level-badge"
- Si es senior: añadir clase "senior" al level-badge
- Si es junior: añadir clase "junior" al level-badge

Archivo: `src/ejercicio6.js`

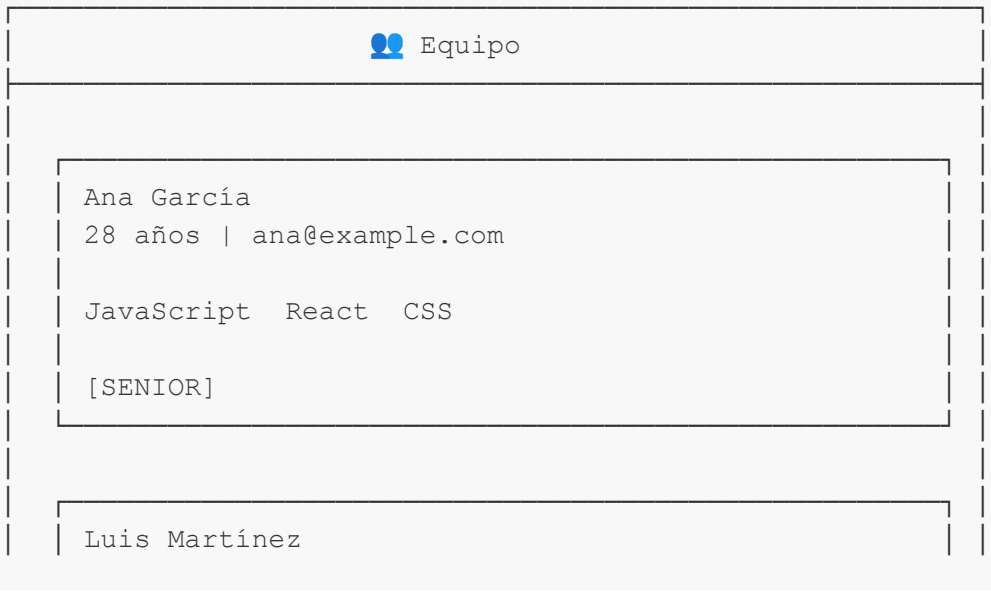
Conceptos que practicarás

- ✅ Crear elementos anidados
- ✅ Manipular arrays dentro de objetos
- ✅ Añadir múltiples clases CSS
- ✅ Crear elementos hermanos
- ✅ Uso de createElement y appendChild

Estructura de datos

```
// El array usuariosConHabilidades contiene:
[
  {
    "id": 1,
    "nombre": "Ana García",
    "edad": 28,
    "email": "ana@example.com",
    "habilidades": ["JavaScript", "React", "CSS"],
    "nivel": "Senior"
  }
]
```

Resultado Esperado



```
│   │ 34 años | luis@example.com                          │ │
│   │                                                     │ │
│   │ HTML  CSS  JavaScript                               │ │
│   │                                                     │ │
│   │ [JUNIOR]                                            │ │
│   └─────────────────────────────────────────────────────┘ │
└─────────────────────────────────────────────────────────────┘
```

🏷️ EJERCICIO 7: Blog de Publicaciones

Nivel: ☆☆☆ Medio

Enunciado

Utiliza el array `publicaciones` para crear un blog con:
Clases CSS OBLIGATORIAS:

- Contenedor principal: "blog-container"
- Cada artículo: "post"
- Título de la publicación: "post-title"
- Metadatos: "post-meta"
- Contenido truncado: "post-content"
- Contenedor de etiquetas: "tags-container"
- Cada etiqueta: "tag"
- Contador de likes: "likes-count"

Archivo: `src/ejercicio7.js`

Conceptos que practicarás

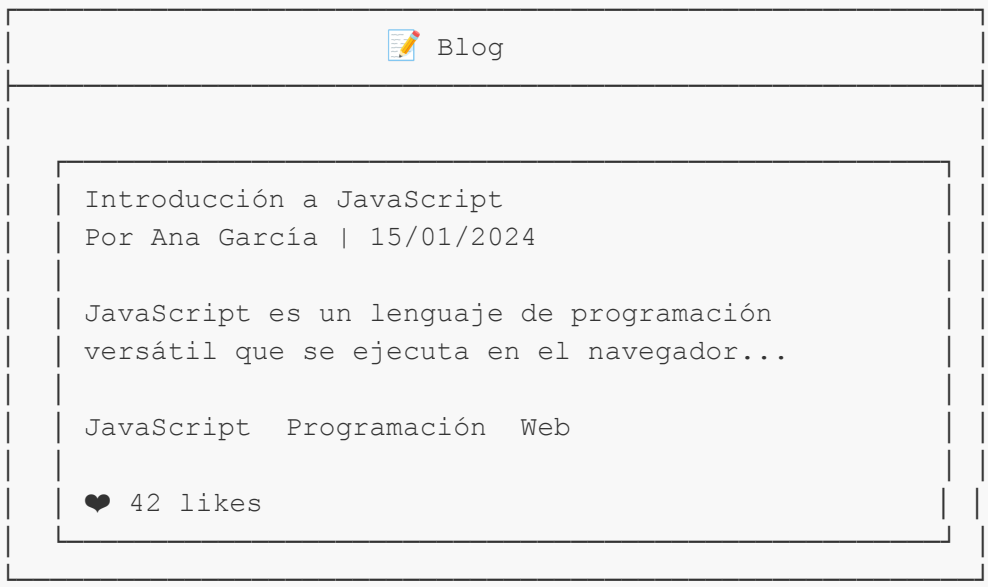
- ✅ Estructura DOM compleja
- ✅ Manipulación de texto (truncado)
- ✅ Creación de elementos interactivos
- ✅ Uso de insertBefore para ordenar elementos
- ✅ Manejo de fechas

Estructura de datos

```
// El array publicaciones contiene:
[
  {
    "id": 1,
    "titulo": "Introducción a JavaScript",
    "autor": "Ana García",
    "fecha": "2024-01-15",
    "contenido": "JavaScript es un lenguaje de programación versátil...",
```

```
    "etiquetas": ["JavaScript", "Programación", "Web"],
    "likes": 42
  }
]
```

Resultado Esperado



🏷 EJERCICIO 8: Menú de Restaurante con Categorías

Nivel: ☆☆☆ Medio

Enunciado

Utiliza el objeto `menuRestaurante` para crear un menú con:
Clases CSS OBLIGATORIAS:

- Contenedor principal: "menu-container"
- Título del restaurante: "restaurant-title"
- Cada sección de categoría: "menu-category"
- Nombre de categoría: "category-title"
- Divisor de categoría: "category-divider"
- Contenedor de platos: "dishes-container"
- Cada plato: "dish-item"
- Contenedor de nombre y precio: "dish-header"
- Nombre del plato: "dish-name"
- Precio: "dish-price"
- Descripción: "dish-description"

Archivo: `src/ejercicio8.js`

Conceptos que practicarás

- ✅ Trabajar con objetos anidados
- ✅ Crear estructuras jerárquicas
- ✅ Uso de nextElementSibling
- ✅ Manipulación de datos numéricos (formateo de precios)
- ✅ Creación de listas anidadas

Estructura de datos

```
// El objeto menuRestaurante contiene:
{
  "categorias": [
    {
      "id": 1,
      "nombre": "Entrantes",
      "platos": [
        { "id": 101, "nombre": "Ensalada César", "precio": 8.50,
"descripcion": "Lechuga romana..." }
      ]
    }
  ]
}
```

Resultado Esperado

```
┌──────────────────────────────────────────────────────┐
│                 🍽 Menú del Restaurante              │
├──────────────────────────────────────────────────────┤
│                                                      │
│  Entrantes                                           │
│  ──────────────────────────────────────────          │
│                                                      │
│  Ensalada César                            €8.50     │
│  Lechuga romana, parmesano, crutones                 │
│                                                      │
│  Bruschetta                                €6.75     │
│  Pan tostado con tomate y albahaca                   │
│                                                      │
│  Gazpacho                                  €5.50     │
│  Sopa fría de tomate y pimiento                      │
│                                                      │
│  Platos Principales                                  │
│  ──────────────────────────────────────────          │
│                                                      │
```

```
│  Paella Valenciana                          €18.90      │
│  Arroz con marisco y pollo                              │
└─────────────────────────────────────────────────────────┘
```

🏷 EJERCICIO 9: Timeline de Eventos

Nivel: ⭐⭐⭐⭐ Medio-Alto

Enunciado

Utiliza el array `eventos` para crear un timeline vertical con:
Clases CSS OBLIGATORIAS:

- Contenedor principal: "timeline-container"
- Línea temporal: "timeline-line"
- Cada evento: "timeline-item"
- Eventos pares: clase adicional "left"
- Eventos impares: clase adicional "right"
- Contenido del evento: "event-content"
- Título del evento: "event-title"
- Fecha y hora: "event-datetime"
- Lugar: "event-location"
- Lista de ponentes: "speakers-list"
- Información adicional: "event-meta"

Archivo: `src/ejercicio9.js`

Conceptos que practicarás

- ✅ Creación de layouts complejos
- ✅ Posicionamiento CSS con DOM
- ✅ Alternado de clases dinámicamente
- ✅ Uso de insertAdjacentElement
- ✅ Manipulación de fechas y horas

Estructura de datos

```
// El array eventos contiene:
[
  {
    "id": 1,
    "nombre": "Frontend Conference 2024",
    "fecha": "2024-03-15",
    "hora": "09:00",
    "lugar": "Centro de Convenciones, Madrid",
    "ponentes": ["Ana García", "Carlos Rodríguez"],
```

```
    "asistentes": 250,
    "precio": 75.00
  }
]
```

Resultado Esperado

💡 Pistas

```
// 1. Importar eventos desde data.js
import { eventos } from '../data.js';

// 2. Crear contenedor principal con clase "timeline-container"
// 3. Crear encabezado <h1> con texto "📅 Timeline de Eventos"
// 4. Crear línea temporal <div class="timeline-line">
// 5. Usar forEach con índice para iterar sobre eventos
// 6. Para cada evento:
//    - Determinar si es par o impar para la posición
//    - Crear <div class="timeline-item left"> o <div class="timeline-item
right">
//    - Crear <div class="event-content"> con toda la información
//    - Crear <h3 class="event-title"> con el nombre
//    - Crear <p class="event-datetime"> formateando fecha y hora
//    - Crear <p class="event-location"> con 📍 y el lugar
```

```
//    - Crear <div class="speakers-list"> con los ponentes
//    - Crear <div class="event-meta"> con precio y asistentes
```

🏷️ EJERCICIO 10: Portafolio de Proyectos con Filtros

Nivel: ⭐⭐⭐⭐⭐ Avanzado

Enunciado

Utiliza el array `proyectos` para crear un portafolio con:
Clases CSS OBLIGATORIAS:

- Contenedor principal: "portfolio-container"
- Contenedor de filtros: "filter-buttons"
- Cada botón de filtro: "filter-btn"
- Botón activo: clase adicional "active"
- Contador de proyectos: "projects-count"
- Grid de proyectos: "projects-grid"
- Tarjeta de proyecto: "project-card"
- Badge de destacado: "featured-badge"
- Contenido de la tarjeta: "project-card-content"
- Título: "project-title"
- Descripción: "project-description"
- Contenedor de tecnologías: "tech-stack"
- Cada tecnología: "tech-badge"

Archivo: `src/ejercicio10.js`

Conceptos que practicarás

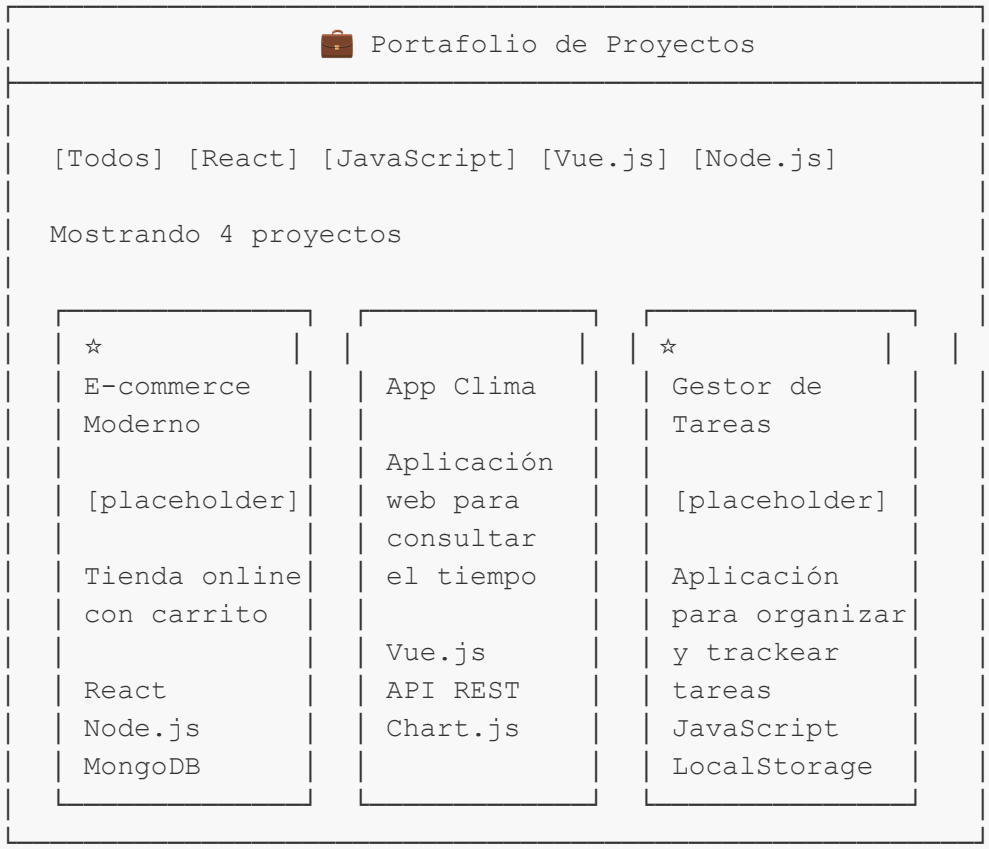
- ✅ Manipulación avanzada del DOM
- ✅ Event handling y filtrado dinámico
- ✅ Creación y eliminación de elementos
- ✅ CSS Grid con JavaScript
- ✅ Estado de la aplicación

Estructura de datos

```
// El array proyectos contiene:
[
  {
    "id": 1,
    "titulo": "E-commerce Moderno",
    "descripcion": "Tienda online con carrito de compras...",
    "tecnologias": ["React", "Node.js", "MongoDB"],
    "imagen": "https://via.placeholder.com/300x200/4f46e5/ffffff?text=E-
```

```
commerce",
    "url": "https://ejemplo-ecommerce.com",
    "destacado": true,
    "fechaFinalizacion": "2024-01-20"
  }
]
```

Resultado Esperado



💡 Pistas

```
// 1. Importar proyectos desde data.js
import { proyectos } from '../data.js';

// 2. Crear contenedor principal con clase "portfolio-container"
// 3. Crear encabezado <h1> con texto "💼 Portafolio de Proyectos"
// 4. Extraer todas las tecnologías únicas de los proyectos
// 5. Crear contenedor de filtros <div class="filter-buttons">
// 6. Crear botón "Todos" y botones para cada tecnología
// 7. Crear contador <div class="projects-count">
// 8. Crear grid de proyectos <div class="projects-grid">
// 9. Función renderProjects(proyectosAFiltrar):
```

```
//     - Limpiar el grid
//     - Iterar sobre proyectosAFiltrar
//     - Para cada proyecto:
//       - Crear <div class="project-card">
//       - Si es destacado, añadir <span class="featured-badge">⭐</span>
//       - Crear imagen <img> con la URL del proyecto
//       - Crear <h3 class="project-title"> con el título
//       - Crear <p class="project-description"> con la descripción
//       - Crear <div class="tech-stack"> con las tecnologías
//       - Para cada tecnología crear <span class="tech-badge">
// 10. Agregar event listeners a los botones de filtro
```

📊 Resumen de los 10 Ejercicios

Ejercicio	Nivel	Conceptos Clave	Líneas
1	⭐	Importación, objetos, createElement	10-15
2	⭐⭐	Arrays, forEach, múltiples elementos	20-30
3	⭐⭐⭐	Objetos, arrays, Grid CSS	35-45
4	⭐⭐⭐	Tablas, formateo de datos	40-50
5	⭐⭐⭐⭐	Formularios, validación, eventos	55-75
6	⭐⭐	Elementos anidados, múltiples clases	45-60
7	⭐⭐⭐	Estructura DOM compleja, manipulación texto	50-70
8	⭐⭐⭐	Objetos anidados, estructuras jerárquicas	55-75
9	⭐⭐⭐⭐	Layouts complejos, posicionamiento	60-85
10	⭐⭐⭐⭐⭐	Manipulación avanzada, estado, eventos	70-100