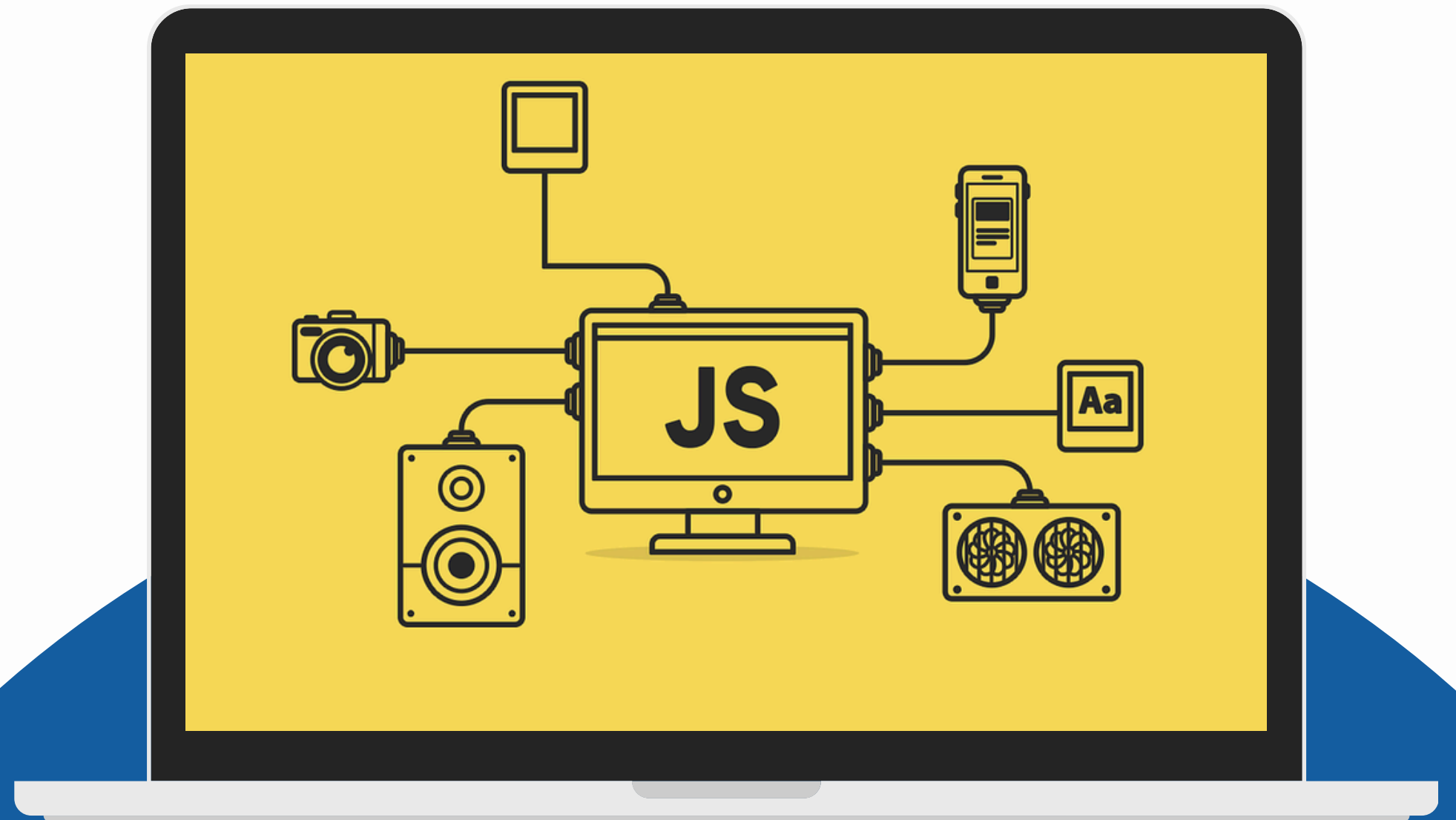


JavaScript Funcional

Alumnos: Martin Jara, Benjamin Pedraza,
Benjamin Diaz



Objetivos del Proyecto

- **Transformar datos CSV usando JavaScript con enfoque funcional.**
- **Aplicar conceptos funcionales.**

Elementos Funcionales Utilizados

- ▶ Iterators y Generators
- ▶ Composición y Pipes
- ▶ Chaining
- ▶ Uso de Lodash
- ▶ Inmutabilidad y Pureza funcional

Diseño de la Solución



The diagram features a large blue semi-circle on the left side. A curved line with four small white circles extends from the right edge of the semi-circle. Two of these circles are highlighted with larger blue circles containing the numbers '01' and '02'. To the right of these circles are the text labels 'Estructura General' and 'Funciones Implementadas' respectively.

01

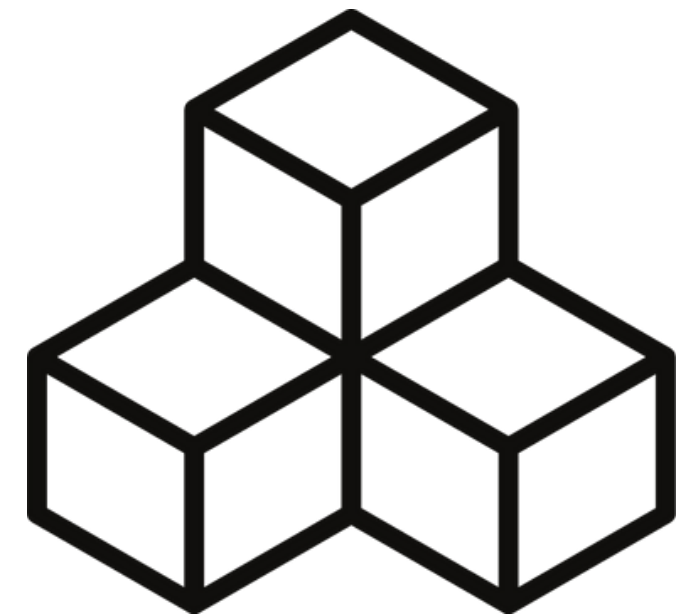
Estructura General

02

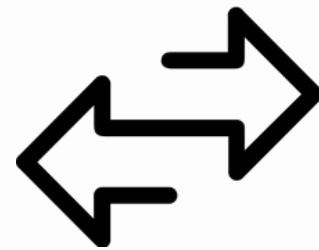
**Funciones
Implementadas**

Estructura General

- **Frontend con Vite + TypeScript:**
 - Interfaz gráfica simple para cargar el CSV y ejecutar las transformaciones.
 - Usa un FileReader para leer el archivo y almacenar su contenido en una matriz de cadenas
- **Módulo de Funciones (Transformaciones CSV):**
 - Funciones puras que reciben y retornan matrices, sin mutar el estado original.



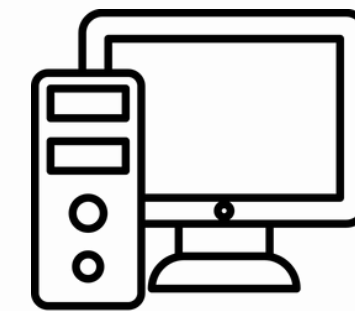
Funciones Implementadas



- Intercambiar columnas.
- Convertir filas a columnas y viceversa.



- Insertar filas y columnas.
- Eliminar filas y columnas.



- Generar tablas HTML.

The background is a blurred office scene with people working at desks, overlaid with a semi-transparent blue layer. Large, light-blue geometric shapes, resembling triangles and polygons, are layered over the blue background. In the center, a white rectangular box with a thin border contains the word "DEMO" in a large, bold, white sans-serif font.

DEMO

Técnicas usadas

- Inmutabilidad: Cada función retorna una nueva matriz sin alterar la original.
- Chaining con Lodash: Operaciones encadenadas para mantener claridad.

```
const createRow = _.flow(  
  (row: string[]) => _.map(row, (dataField) => `\t\t<td>${dataField}</td>`),  
  (dataFields: string[]) => `\t<tr>\n${_.join(dataFields, '\n')}\n\t</tr>`  
);
```



```
=> _.map(row, (dataField) => `
ing[])) => `\\t<tr>\\n${_.join(dat
```

Generación de HTML Funcional y Limpia

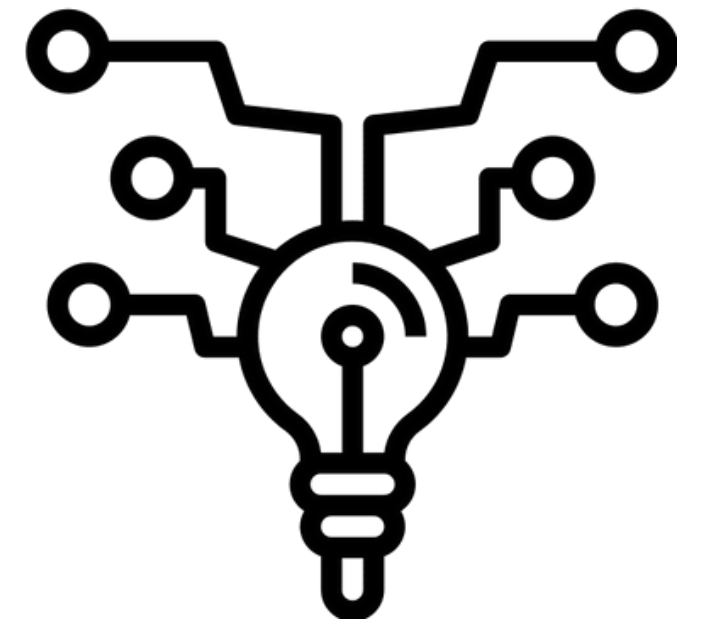
- Uso de `_.map`, `_.join` y composición funcional para construir HTML.

Manejo seguro de CSV

- Validación en operaciones para prevenir errores en índices fuera de rango.

Aplicacion de Conceptos

- Chaining con Lodash para simplificar operaciones.
- Iterators y Generators: potencial uso futuro para manejar grandes archivos CSV.
- Manejo Asíncrono: preparación del código para posibles futuras extensiones asincrónicas usando promesas



Documentación y Mantenibilidad

- README detallado: Instalación, ejecución y uso claro.
- Código documentado y estructurado de forma modular.



**Muchas
gracias!**

