Manual Técnico del Proyecto OakLand

1. Descripción General

Este proyecto, **OakLand**, es un intérprete basado en un lenguaje de programación personalizado, implementado utilizando JavaScript puro. La interfaz gráfica permite escribir, cargar y ejecutar archivos ".oak", además de generar reportes de errores y de la tabla de símbolos.

2. Tecnologías Utilizadas

- **JavaScript Puro**: El núcleo del proyecto está desarrollado completamente en JavaScript, sin el uso de frameworks o bibliotecas adicionales. Se utilizaron funciones nativas de JavaScript para gestionar la lógica del lenguaje, así como para interactuar con el DOM (Document Object Model) de la interfaz.
- **PEG.js** (**Parser Expression Grammar**): Se utilizó **PEG.js** para la creación del **parser**. PEG.js es una herramienta que permite generar analizadores sintácticos a partir de gramáticas formales. En este caso, la gramática definida permite procesar las instrucciones del lenguaje OakLand, convertirlas en árboles de sintaxis abstractos (ASTs), y entregarlas al intérprete para su ejecución.
- **Patrón Visitor**: Para el manejo del procesamiento de las instrucciones, se implementó el patrón de diseño **Visitor**, que permite ejecutar comportamientos sobre nodos del AST sin modificar la estructura de dichos nodos. Este patrón fue clave para gestionar:
 - o Variables: Declaración, asignación, y manejo de tipos.
 - o **Arreglos**: Inicialización, manipulación, y acceso a sus elementos.
 - o **Sentencias**: Condicionales, bucles, y expresiones.
 - Clases y Objetos: Definición de clases, creación de instancias, y manejo de atributos y métodos.

3. Estructura del Proyecto

El proyecto está dividido en varios componentes clave:

- **Parser**: El archivo de gramática generado por **PEG.js** transforma el código fuente .oak en un AST. Este árbol es procesado por el intérprete a través de visitas a cada nodo, aplicando la lógica definida en el patrón **Visitor**.
- **Interprete**: Implementado en JavaScript, este componente recibe el AST del parser y lo recorre utilizando el patrón Visitor. Aquí es donde se gestionan las declaraciones, asignaciones, y ejecución de las distintas instrucciones del lenguaje.
- Interfaz Gráfica (HTML/CSS): La interfaz fue desarrollada usando HTML y CSS, ofreciendo una experiencia visual simple pero funcional. El diseño fue optimizado para la entrada y salida de comandos, la carga de archivos, y la generación de reportes. CSS se usó para dar estilo a los botones, áreas de texto y consola, manteniendo una estética minimalista.

4. Detalles de Implementación

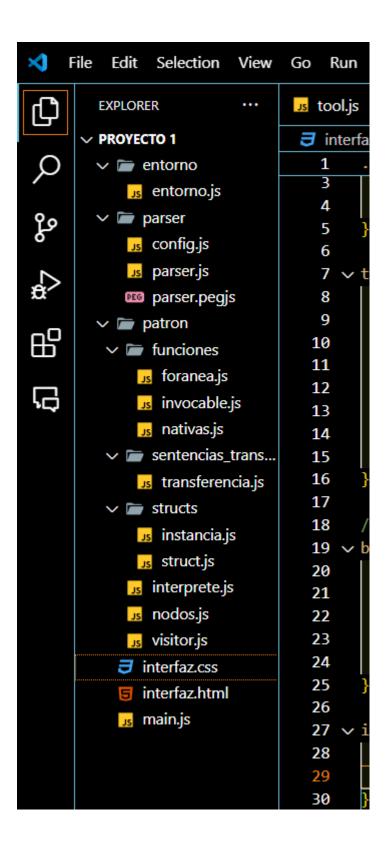
- Manejo de Variables y Tipos: Las variables pueden ser de diferentes tipos, como int, float, string, boolean, y char. Las variables se manejan dentro de un entorno dinámico que valida los tipos y sugiere errores cuando se intenta asignar un tipo incorrecto a una variable.
- **Arreglos**: Se implementaron arreglos con soporte para múltiples tipos. El sistema valida que los elementos dentro del arreglo sean consistentes con el tipo declarado, permitiendo manipular datos como listas o colecciones ordenadas.
- Clases y Objetos: El proyecto soporta la definición de clases con atributos y métodos, siguiendo el paradigma orientado a objetos. Los objetos creados a partir de clases pueden interactuar mediante invocaciones de métodos y acceso a atributos.
- Sentencias de Control: Se implementaron las sentencias condicionales (if, else) y los ciclos (while, for), permitiendo la ejecución dinámica de bloques de código en función de ciertas condiciones o iteraciones.

5. Despliegue

El proyecto está diseñado para ser desplegado utilizando **GitHub Pages**. Esto permite que la aplicación web sea accesible desde cualquier navegador sin necesidad de instalación local. El código fuente, incluyendo el parser, el intérprete, y la interfaz, está completamente alojado en un repositorio de GitHub.

6. Conclusión

El proyecto **OakLand** combina el poder de **JavaScript puro** con la flexibilidad de **PEG.js** y el **patrón Visitor** para ofrecer una experiencia completa de interpretación de código a través de una interfaz web. El despliegue en **GitHub Pages** asegura que el proyecto sea fácilmente accesible y mantenible.



```
containe {
    width: 600px;
    margin: 0 auto;
    text-align: center;
}

textarea {
    width: 600px;
    height: 400px;
    margin-botton: 20px;
    background-color: __black;
    color: __skyblue;
    border: px solid __rgb(246, 255, 0);
    border-radius: 5px;
    font-size: large;
}

/* AJUSTE DE SIZE DE AMBOS BOTONES */
button {
    font-size: 18px; /* Aumenta el tamaño del texto dentro del botón */
    padding: 10px 20px; /* Añade relleno para hacer que el botón sea más grande */
    width: 150px; /* Establece un ancho fijo */
    border: 3px solid __rgb(0, 217, 255);
    cursor: pointer; /* Cambia el cursor a una mano cuando se pasa por encima */
}

input {
    font-size: 18px;
    width: 170px;
}

.botones{
    text-align: left;
```

```
| ClDCTYPE html>
| chtml lang="em">
| chtml lang="e
```

DIEGO CHEN 202202882