Manual Tecnico

Detalles tecnicos:

Aplicacion web desarrollada empleando JavaScript con ayuda de los frameworks: JQuery, Bootstrap y Jison. El IDE empleado para el desarrollo fue: Webstorm. Sistema operativo empleado durante el desarrollo: Linux. Navegador principal empleado para la mayoria de pruebas: FireFox.

Como correr:

Es una aplicacion web implementada totalmente del lado del cliente, no hay necesidad de emplear ningun procedimiento especial para correrla, simplemente abra la pagina: index.html en cualquier navegador para comenzar.

Descripcion del source code:

El source code de la aplicacion se encuentra disperso en varias carpetas. A continuacion se listan las carpetas que continen scripts de liberias:

- modes

  Carpeta que contiene todos los scripts necesarios para el syntax colouring de Java.

- lib

  Carpeta que contiene los scripts necesarios para la libreria CodeMirror

- assets/js

  Carpeta que contiene todo el source code de la aplicacion. Contiene tanto los scripts de JQuery como los scripts que yo mismo implemente para la logica del sistema.

- bootstrap/js

  Carpeta que contiene scripts especiales para el funcionamiento del framework bootstrap.

Descripcion de los archivos importantes del source code:

A continuacion se lista un quick-overview de todos los archivos de source code empleados por la aplicacion:

- “_3D_grammar.js”

  Script generado por Jison empleando el archivo de gramatica “_3D_grammar.jison” ubicado en la carpeta 3D_Grammar. Este script es el encargado de parsear la entrada de 3D y llenar las estructuras de ejecucion como lo son: La lista de labels y la lista de instrucciones.

- “_Optimizer_grammar.js”

  Script generado por Jison empleando el archivo de gramatica “_Optimizer_grammar.jison” ubicado en la carpeta Optimizer_Grammar. Este script es el encargado de parsear la entrada de 3D y llenar estrucuturas especiales para la optimizacion de codigo. Como lo son: La lista de instrucciones y la lista de Instrucciones no usadas.

- “_Import_Grammar.js”

  Script generado por Jison empleando el archivo de gramatica “_Import_grammar.jison” ubicado en la carpeta Import_Grammar. Este script es el encargado de parsear la entrada de alto nivel con varios imports y resolverlos para generar una sola entrada unificada.

- “_Aux_Grammar.js”

  Script generado por Jison empleando el archivo de gramatica “_Aux_Grammar.jison” ubicado en la carpeta Aux_Grammar. Este script es el encargado de parsear la entrada unificada y resolver todas las clases del programa. Esta gramatica no verifica la sintaxis de Yaba como tal directamente, sino unicamente la lista de clases errores sintacticos en esta fase podrian ser por ejemplo dejar un brace abierto. El objetivo de esta gramatica es unicamente hacer un pre-analisis sobre todas las clases del programa.

- “_Y_Grammar.js”

  Script generado por Jison empleando el archivo de gramatica “_Y_Grammar.jison” ubicado en la carpeta Y_Grammar. Este script es el encargado de parsear la entrada unificada de Yaba y generar el AST. Este script es el que verifica toda la sintaxis propia del lenguaje.

- “_3D_Utility.js”

  Script que contiene la declaracion e implementacion de varias constantes o funciones globales de utilidad durante toda la interpretacion del codigo.

- “_3D_Interpreter.js”

  Script que detalla todas las directivas del programa, tambien implementa directamente el interprete del codigo 3D.

- “_Y_Compiler_Utility.js”

  Script que contine la declaracion e implementacion de varias constantes y funciones de uso general y mas que todo tienen impacto directo en la interaccion con el usuario y la recopilacion/muestra de informacion. Algunos metodos son por ejemplo: sorting, addFolder. El primero es un objeto constante que tiene por objetivo implementar varios metodos de ordenamiento, por el momento el metodo de ordenamiento empleado por la aplicacion es el merge sort. El segundo es un metodo de muestra que agrega una nueva pestana al editor de texto en la ventana principal.

- “Compiler_Utility.js”

  Script que contiene la declaracion e implementacion de todos los metodos empleados por el compilador durante ambas etapas: La compilacion y la generacion de codigo. Este script se encarga de la extraccion del codigo fuente y de la llamada a los diferentes parsers segun sea el caso y la etapa. Ademas controlar el flujo de la aplicacion, tambien define varios metodos y constantes de utilidad durante las dos etapas siguientes. Como lo es el metodo: perform_inheritance() por ejemplo.

- “Printing”

  Este escript define una constante Printing. Que se encarga de escribir directamente el codigo intermedio, guardarlo y finalmente publicarlo en la tab de ejecucion al finalizar el proceso.

- “Y_Compiler.js”

  Este script define una constante importante: Compiler. Esta constante es la que se encarga directamente de procesar el AST y construir tanto la tabla de simbolos como la lista de clases. Todo este proceso se encuentra encapsulado en este archivo.

- “Code_Generator.js”

  Este script define una constante importante: Code_Generator. Esta constante es la que se encarga de recoger la informacion obtenido por el Compiler en la etapa anterior (obtiene el AST, la tabla de simbolos y la tabla de clases del Compiler). Y posteriormente las emplea para generar el codigo intermedio. Todo el proceso de generacion de codigo intermedio se encuentra encapsulado en este archivo.