Gramática del Analizador Sintáctico

Procesador JavaScript-PDL

Serrano,Arrese Francisco Javier
Cañibano,Lopez Alberto
Vallejo,Collados Jesús

Grupo 14
Procesadores de Lenguajes
Universidad Politécnica de Madrid
Curso 2020-2021

Gramática Analizador Sintáctico

```
No terminales:
{
    Main, Programa, Funcion, Cuerpo, Sentencia, Bloque, Exprexion, Condicion, Condicion2,
    Aritmetica, Types, ParametrosFun, ParametrosFun2, Tipo
}

Axioma: Main

Terminales:
{
alert,boolean, else, function, if, input, let, number, return, string, while, false, true, do,
autoInc, Número, Posición(Número), Número, equal, colon, semicolon, openPar, closePar, openBraq,
closeBraq, plus, minus, and, not, notEquals, equals
}

Producciones = {

    MAIN -> PROGRAMA PROGRAMA      // Siempre tiene que abrirse un programa (como minimo)

                                   //  ---- Puedo llamar: ----
    PROGRAMA -> CUERPO PROGRAMA    // A un cuerpo de porgrama
    PROGRAMA -> FUNCION PROGRAMA   // A una funcion
    PROGRAMA ->                    // O terminar


    FUNCION -> function id openPar PARAMETROSFUN closePar openBraq  CUERPO closeBraq


//  ---- Dentro del cuerpo podemos ----
// Definir una varible
// hacer if simples
// hacer if de una sola linea (sin corchetes)
// hacer un do While

    CUERPO -> let TIPO id semicolon
    CUERPO -> if  openPar CONDICION closePar openBraq BLOQUE closeBraq
    CUERPO -> if  openPar CONDICION closePar SENTENCIA semicolon
    CUERPO -> do openBraq BLOQUE closeBraq while  openPar CONDICION  closePar

    //  ---- Podemos declara sentecias ----
    SENTENCIA -> id  igual  EXPRESION                  // identificador = expresion
    SENTENCIA -> id  openPar  PARAMETROSFUN  closePar  // llamamos a una funcion con sus parametros
    SENTENCIA -> id alert openPar EXPRESION closePar   // Crea una alerta
    SENTENCIA -> return  RETURNVALUE                   // devolveria un returnvaule

    //  ---- Con esto podemos encadenar ----
    BLOQUE -> CUERPO                   // Por un lado encadenar los if y las cosas de dentro
    BLOQUE -> SENTENCIA BLOQUE         // Con en esto podemos encadenar sentencias
```

```
BLOQUE ->                          // Terminamos

 // ---- Posibles expresiones ----
EXPRESION -> ARITMETICA semicolon  // una operacion aritmetica y ;
EXPRESION -> not TYPES semicolon   // una negacion de un TYPE y ;
EXPRESION -> TYPES semicolon       // un TYPES a secas
EXPRESION ->                       // o terminar

//  ---- Posibles condiciones ----
CONDICION -> TYPES notequals TYPES CONDICION2      // Puede ser diferente
CONDICION -> TYPES equals TYPES CONDICION2         // Puede ser igual
CONDICION2 -> and CONDICION                        // Pueden encadenarse varias condiciones
CONDICION2 ->                                      // Podemos terminar

// ---- Operaciones que se puede hacer a los id ----
ARITMETICA -> TYPES plus TYPES ARITMETICA   // a + b
ARITMETICA -> TYPES minus TYPES ARITMETICA  // a - b
ARITMETICA -> TYPES autoInc                 // a++

// ---- Elementos de entradas de una expresion ----
TYPES-> id     // identificador
TYPES-> ent    // entero
TYPES-> cad    // cadena
TYPES -> true  // Verdadero
TYPES -> false // Falso
TYPES-> id openPar  LLAMADAFUN  closePar semicolon // LLamada a una funcion

//  ---- Como tratamos los datos que introduciomos auna funcion ----
PARAMETROSFUN -> id  PARAMETROSFUN2 // Tiene que tener un id como minimo
PARAMETROSFUN2 -> colon PARAMETROSFUN // para encadenar usaremos las comas
PARAMETROSFUN2 ->  // y si queremos terminar salimos

//  ---- Tipos de datos que podemos tener ----
TIPO -> string
TIPO -> number
TIPO -> boolean
```