T1: Linguagens de Programação

Thiago Carreira A. Nascimento
thiago.nascimento@acad.pucrs.br

Giuseppe Menti
mentifg@gmail.com

March 25, 2018

Abstract

O presente trabalho visa demonstrar uma implementação de um validador de sintaxe para a Linguagem *Java Script* com **XText**[**?**]. Para tanto, foi escolhido um único comando, e, a partir da sua sintaxe, foi desenvolvida uma gramática que a reconhece. O comando escolhido foi **var**[**?**], que indica a declaração de uma varíavel.

1 Regras para utilização do comando var em *JavaScript*

As regras prescritas par ao uso do comando **var** em *Java Script* de acordo com [] são:

2 Desenvolvimento: nosso validador de sintaxe

```
grammar org.xtext.example.mydsl.T1 with
    org.eclipse.xtext.common.Terminals

generate t1 "http://www.xtext.org/example/mydsl/T1"

Model:
vars+=Var*;

Name:
ID
;

Value:
ID | STRING | INT| INT? '.' INT* | INT '+' INT | INT '+' INT? '.'
    INT* | INT '+' INT* | INT '-' INT |
INT '-' INT? '.' INT* | INT '-' INT* | INT? '.' INT* '+' INT | '-'
    INT* |
('-' INT)* | '-' INT* '+''.'? INT* | '-' INT '-' INT? '.'INT*
```

```
;

Attribution:
('=' Value)*
;

Declaration:
Name Attribution
;
Var:
'var' declarations+=Declaration(',' declarations+=Declaration)*(';')?;
```

3 Testes Unitários Realizados

Foram realizados no total de 20 testes unitários a fim de validar a sintaxe para o comando **var** em *Java Script*. Eis os métodos:

```
@Test
def void validaAtribuicaoComSubtracao() {
val result = parseHelper.parse('''
var a = 1 - 2;
''')
Assert.assertNotNull(result)
Assert.assertTrue(result.eResource.errors.isEmpty)
}

@Test
def void validaAtribuicaoComSubtracaoFloat() {
val result = parseHelper.parse('''
var a = 1 - 3.2121;
''')
Assert.assertNotNull(result)
Assert.assertTrue(result.eResource.errors.isEmpty)
}

@Test
def void validaAtribuicaoComNegacaoESubtracaoFloat() {
val result = parseHelper.parse('''
var a = -1 - 3.2121;
''')
Assert.assertNotNull(result)
Assert.assertTrue(result.eResource.errors.isEmpty)
}

@Test
def void validaAtribuicaoComSoma() {
val result = parseHelper.parse('''
var a = 1 + 2;
```

```
''')
Assert.assertNotNull(result)
Assert.assertTrue(result.eResource.errors.isEmpty)
}

@Test
def void validaAtribuicaoComSomaFloat() {
val result = parseHelper.parse('''
var a = 1 + 2.2121;
''')
Assert.assertNotNull(result)
Assert.assertTrue(result.eResource.errors.isEmpty)
}

@Test
def void validaVarFloatSemValorEsquerdaSoma() {
val result = parseHelper.parse('''
var a = .12312312312 + 111;
''')
Assert.assertNotNull(result)
Assert.assertTrue(result.eResource.errors.isEmpty)
}

@Test
def void validaAtribuicaoComum() {
val result = parseHelper.parse('''
var a = b;
''')
Assert.assertNotNull(result)
Assert.assertTrue(result.eResource.errors.isEmpty)
}

@Test
def void vaidaAtribuicaoComposta() {
val result = parseHelper.parse('''
var a = 0, b = 2;
''')
Assert.assertNotNull(result)
Assert.assertTrue(result.eResource.errors.isEmpty)
}

@Test
def void validaVarSemPontoVirgula() {
val result = parseHelper.parse('''
var a = c
''')
Assert.assertNotNull(result)
Assert.assertTrue(result.eResource.errors.isEmpty)
}
```

```
@Test
def void validaVarAspasSimples() {
val result = parseHelper.parse('''
var a = ' 123**& ';
''')
Assert.assertNotNull(result)
Assert.assertTrue(result.eResource.errors.isEmpty)
}

@Test
def void validaVarAspasDuplas() {
val result = parseHelper.parse('''
var a = "''"
''')
Assert.assertNotNull(result)
Assert.assertTrue(result.eResource.errors.isEmpty)
}

@Test
def void validaVarSemAtribuicao() {
val result = parseHelper.parse('''
var a
''')
Assert.assertNotNull(result)
Assert.assertTrue(result.eResource.errors.isEmpty)
}

@Test
def void validaVarSemAtribuicaoComVirgula() {
val result = parseHelper.parse('''
var a;
''')
Assert.assertNotNull(result)
Assert.assertTrue(result.eResource.errors.isEmpty)
}

@Test
def void validaVarAtribuicaoNumero() {
val result = parseHelper.parse('''
var a = 12312312312.122;
''')
Assert.assertNotNull(result)
Assert.assertTrue(result.eResource.errors.isEmpty)
}

@Test
def void validaVarFloatSemDefinicaoCasaDecimal() {
val result = parseHelper.parse('''
var a = 12312312312.;
''')
```

```
Assert.assertNotNull(result)
Assert.assertTrue(result.eResource.errors.isEmpty)
}

@Test
def void validaVarDuplamenteComposta() {
val result = parseHelper.parse('''
var a, b = a = 'A';
''')
Assert.assertNotNull(result)
Assert.assertTrue(result.eResource.errors.isEmpty)
}


@Test
def void validaVarFloatSemValorEsquerda() {
val result = parseHelper.parse('''
var a = .12312312312
''')
Assert.assertNotNull(result)
Assert.assertTrue(result.eResource.errors.isEmpty)
}

@Test
def void validaVarInicioNumero(){
val result = parseHelper.parse('''
var 12311 = a;
''')
Assert.assertNotNull(result)
Assert.assertFalse(result.eResource.errors.isEmpty)
}

@Test
def void validaVarInicioCaracter(){
val result = parseHelper.parse('''
var &asa12 = 12;
''')
Assert.assertNotNull(result)
Assert.assertFalse(result.eResource.errors.isEmpty)
}

@Test
def void validaVarCaracterEspecial(){
val result = parseHelper.parse('''
var varC7123123 = *****;
''')
Assert.assertNotNull(result)
Assert.assertFalse(result.eResource.errors.isEmpty)
}
```