الجمهورية العربية السورية
جامعة دمشق
كلية الهندسة المعلوماتية

مشروع مترجمات للغة Angular

مشروع أعد للقسم العملي من مادة المترجمات للسنة الرابعة – قسم هندسة البرمجيات

إعداد الطلاب:

محمد رياض عمر

مجد جميل لوقا

مجد سامرعروس

بثينه محمود السبيريج

إشراف:

د.باسم قصيبة – م.هدى الحوامدة

What is **ANTLR**?

ANTLR(ANother Tool For Language Recognition) هو مولد تحليل يُستخدم لإنشاء مترجمات ومحللات للغات برمجة وتنسيقات ملفات مختلفة.

يتيح ANTLR للمطورين تحديد قواعد لغوية(Grammars) باستخدام قواعد تحليلية مبسطة، ثم يُنشئ ANTLR محللاً(Parser) ومولداً للمحلل(Lexer)يستند على هذه القواعد.

في سياق مترجمات اللغات البرمجية:

Lexer(محلل الأقسام):يقوم بتحويل سلسلة من الرموز(tokens) من المدخلات إلى وحدات أساسية، مثل الكلمات المفتاحية، المعاملات، والرموز الترقيمية.

Parser(محلل الجمل):يقوم بفحص الترتيب اللغوي للرموز المُحللة من قبل المحلل، ويبني بنية بيانات تُعبر عن تركيب البرنامج وعلاقات بين مكوناته.

يقوم ANTLR بتوليد كود في لغة البرمجة المستهدفة(مثل جافا) للمحلل والمولد على أساس القواعد التي يتم تحديدها. القواعد تُعبر عن قاعدة اللغة المراد تحليليها.

مميزات ANTLR تشمل:

سهولة الاستخدام: تعتبر قواعد ANTLR أن تكون أكثر قرباً إلى الصياغة اللغوية المألوفة، مما يجعل من السهل على المطورين تحديد قواعد لغة جديدة.

دعم للغات متعددة: يمكن استخدام ANTLR لتحليل لغات متنوعة، سواء كانت لغات برمجة أو تنسيقات ملفات أخرى.

تكامل مع أنظمة متعددة: يمكن توليد المحللات باستخدام ANTLR لتناسب مجموعة متنوعة من الأنظمة والمشاريع.

مجتمع نشط: لديها مجتمع نشط من المستخدمين والمطورين، ويتم تحديث الأداة بانتظام.

What is **Angular**?

Angular هي إطار عمل مفتوح المصدر يُستخدم لتطوير تطبيقات ويب حديثة وأحادية الصفحة (Single Page Applications - SPA). تم تطويره وصيانته بواسطة فريق Google ويعتمد على لغة TypeScript التي تعتبر امتدادًا للغة JavaScript.

Angular يُسهّل بناء تطبيقات ويب ديناميكية، حيث يوفر أدوات متقدمة مثل مكونات قابلة لإعادة الاستخدام، التوجيه (Routing)، وحقن التبعية (Dependency Injection). يتميز بقوته وأدائه العالي في التطبيقات الكبيرة والمعقدة، مما يجعله خيارًا شائعًا بين المطورين.

بعض المفاهيم في **Angular**:

Modules (الوحدات):

هي أجزاء أساسية تنظم الكود في Angular، حيث يتم تجميع المكونات والخدمات والتوجيهات المتعلقة ببعضها في وحدات مستقلة.

Components (المكونات):

كل تطبيق Angular يتكون من مكونات. المكون يتكون من ملف TypeScript يحتوي على منطق العمل، وملف HTML يمثل واجهة المستخدم، وملف CSS للتنسيق.

Templates (القوالب):

تستخدم لتحديد واجهة المستخدم، حيث يتم الجمع بين HTML مع توجيهات Angular مثل *ngIf و*ngFor لإضافة السلوك التفاعلي.

Directives (التوجيهات):

أدوات تُستخدم لتوسيع سلوك عناصر DOM، مثل إنشاء عناصر ديناميكية أو تطبيق منطق مخصص.

Services (الخدمات):

تُستخدم لمشاركة المنطق أو البيانات بين المكونات، حيث يتم إنشاء الخدمات ليتم حقنها عند الحاجة.

Dependency Injection (حقن التبعية):

نمط تصميم يُستخدم في Angular لتزويد المكونات والخدمات بما تحتاجه من تبعيات بشكل تلقائي.

Routing (التوجيه):

نظام لإدارة التنقل بين صفحات التطبيق باستخدام عناوين URL.

Two-way Data Binding (ربط البيانات ثنائي الاتجاه):

يتيح التزامن الفوري بين واجهة المستخدم وبيانات النموذج.

Pipes (الأنابيب):

تُستخدم لتحويل البيانات وعرضها بتنسيقات معينة في القوالب (مثل تحويل النصوص إلى حروف كبيرة أو تنسيق التواريخ).

Lifecycle Hooks (خطافات دورة الحياة):

مجموعة من الأحداث التي يمكن أن يتفاعل معها المطور أثناء دورة حياة المكون، مثل ngOnInit وngOnDestroy.

Observables (القابلات للملاحظة):

جزء من مكتبة RxJS تُستخدم للتعامل مع البيانات غير المتزامنة مثل طلبات HTTP أو الأحداث.

الهدف من المشروع:

بناء مترجم لتحويل قواعد Angular إلى كود صالح للتنفيذ على المتصفح وذلك عبر مراحل عديدة نكتفي بها للقسم العملي لهذا الفصل(التحليل اللفظي واللغويLexical Analysis –التحليل القواعدي Syntactical Analysis - بناء شجرة AST- Visitor Functionc – بناء جدول Symbol Table).يتم تحقيق ذلك من خلال فهم قواعد كتابة Angular،التي تُستخدم لبناء وتنظيم واجهات المستخدم بشكل متكامل وفعّال.

التحليل اللفظي واللغوىLexical Analysis:

هي المرحلة الأولى عندما يقوم المترجم بمسح الكود المصدري من اليسار لليمين، حرفاً بحرف، وتجميع هذه الأحرف في الرموز المميزة.

الوظائف الأساسية لهه المرحلة هي:

- تحديد الوحد
- ات المعجمية في الكود المصدري.
- تصنيف الوحدات المعجمية إلى فئات مثل الثوابت والكلمات المحجوزة، وإدخالها في جداول مختلفة،مع العلم أنه سيتجاهل التعليقات في البرنامج المصدري.
- تحديد الرمز الذي ليس جزءاً من اللغة.

إن أداة **Lexer** وظيفتها التعرف على تسلسل الأحرف بتمثيل هذا التسلسل برمز Token.حيث تعرف الرموز بحروف كبيرة ضمن ملف ال Lexer، وترتيب تعريف الرموز مهم جداً في هذه المرحلة ولا يمكن تعريف أكثر من Keyword بنفس الاسم بها.

في مشروعنا نهتم بأهم الأساسيات الخاصة بلغة Angular والتي يتم تعريفها في ملف Lexer:

١.تعريف الكلمات المحجوزة الخاصة بتعريف الثوابت والمتغيرات والاستيراد والتصدير:

٢.تعريف الكلمات المحجوزة بهياكل التحكم والحلقات Loops and Control Flow

٣.تعريف الكلمات المحجوزة الخاصة بالصفوف Classesوالبرمجة الكائنية OOP:

٤.تعريف أوامر الطباعة:

٥.تعريف التوابع:

٦.تعريف العمليات بأنواعها:

- عمليات المقارنة:
- العمليات الرياضية:
- العمليات المنطقية:
- عمليات الإسناد والزيادة والنقصان:

٧. تحليل السلاسل النصية:

٨.تعريف المتحولات ضمن سلسلة نصية:

٩.التعامل مع الأعداد:

١٠.تعريف علامات الترقيم:

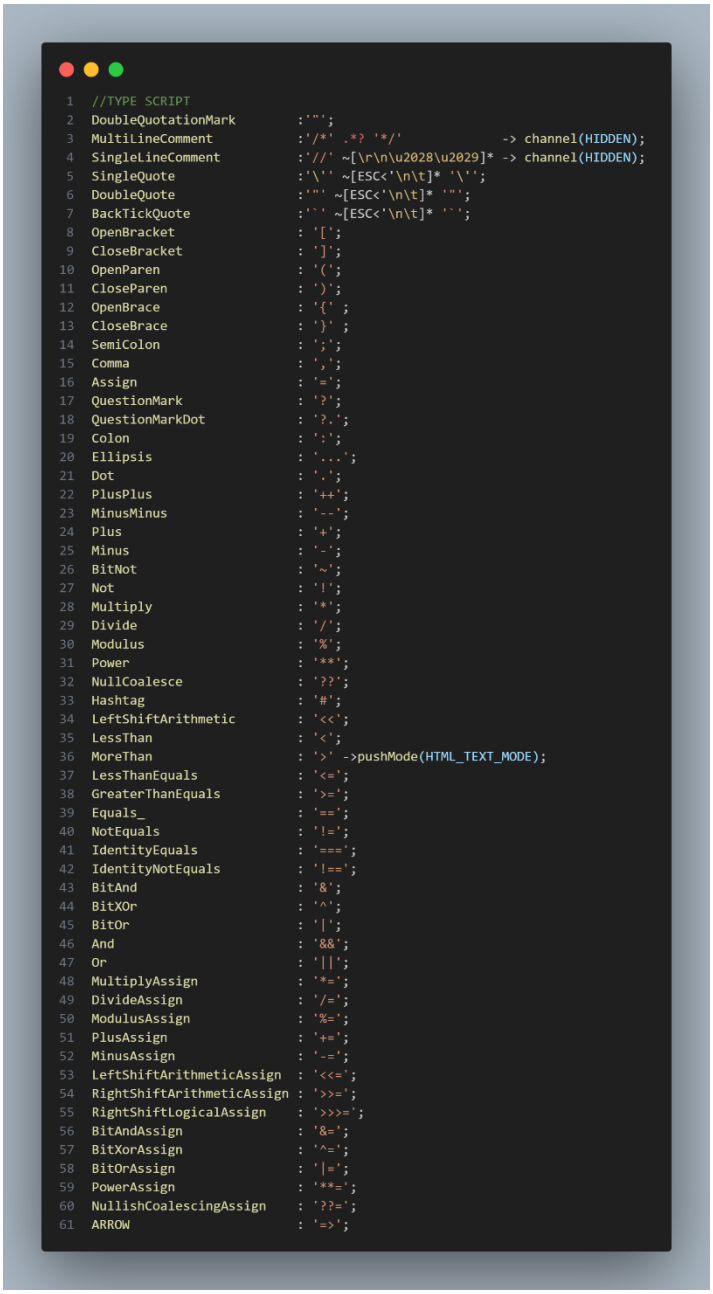
١١.تعريف الأقواس بمختلف أنواعها:

١٢.المساحات البيضاء والأسطر الجديدة والتعليقات:

```
DoubleQuotationMark     :'"';
MultiLineComment        :'/*' .*? '*/'              -> channel(HIDDEN);
SingleLineComment       :'//' ~[\r\n\u2028\u2029]* -> channel(HIDDEN);
SingleQuote             :'\'' ~[ESC<'\n\t]* '\'';
DoubleQuote             :'"' ~[ESC<'\n\t]* '"';
BackTickQuote           :'`' ~[ESC<'\n\t]* '`';
```

```
/// Keywords

Break      : 'break';
Do         : 'do';
Instanceof : 'instanceof';
Typeof     : 'typeof';
Case       : 'case';
Else       : 'else';
New        : 'new';
Var        : 'var';
Catch      : 'catch';
Finally    : 'finally';
Return     : 'return';
Void       : 'void';
Continue   : 'continue';
For        : 'for';
Switch     : 'switch';
While      : 'while';
Debugger   : 'debugger';
Function_  : 'function';
This       : 'this';
With       : 'with';
Default    : 'default';
If         : 'if';
Throw      : 'throw';
Delete     : 'delete';
In         : 'in';
Try        : 'try';
As         : 'as';
From       : 'from';
ReadOnly   : 'readonly';
Async      : 'async';
Await      : 'await';
Yield      : 'yield';
YieldStar  : 'yield*';
```

```
Class   : 'class';
Enum    : 'enum';
Extends : 'extends';
Super   : 'super';
Const   : 'const';
Export  : 'export';
Import  : 'import';

/// The following tokens are also considered to be FutureReservedWords
/// when parsing strict mode

Implements : 'implements';
Let        : 'let';
Private    : 'private';
Public     : 'public';
Interface  : 'interface';
Package    : 'package';
Protected  : 'protected';
Static     : 'static';

//keywords:
Any        : 'any';
Number     : 'number';
Never      : 'never';
Boolean    : 'boolean';
String     : 'string';
Int        : 'int';
Unique     : 'unique';
Symbol     : 'symbol';
Undefined  : 'undefined';
Object     : 'object';

Of      : 'of';
KeyOf   : 'keyof';

TypeAlias: 'type';

Constructor : 'constructor';
Namespace   : 'namespace';
Require     : 'require';
Module      : 'module';
Declare     : 'declare';
Abstract    : 'abstract';
Is          : 'is';
At          : '@';
```



```
//TYPE SCRIPT
DoubleQuotationMark        :'"';
MultiLineComment           :'/*' .*? '*/'             -> channel(HIDDEN);
SingleLineComment          :'//' ~[\r\n\u2028\u2029]* -> channel(HIDDEN);
SingleQuote                :'\'' ~[ESC<'\n\t]* '\'';
DoubleQuote                :'"' ~[ESC<'\n\t]* '"';
BackTickQuote              :'`' ~[ESC<'\n\t]* '`';
OpenBracket                : '[';
CloseBracket               : ']';
OpenParen                  : '(';
CloseParen                 : ')';
OpenBrace                  : '{' ;
CloseBrace                 : '}' ;
SemiColon                  : ';';
Comma                      : ',';
Assign                     : '=';
QuestionMark               : '?';
QuestionMarkDot            : '?.';
Colon                      : ':';
Ellipsis                   : '...';
Dot                        : '.';
PlusPlus                   : '++';
MinusMinus                 : '--';
Plus                       : '+';
Minus                      : '-';
BitNot                     : '~';
Not                        : '!';
Multiply                   : '*';
Divide                     : '/';
Modulus                    : '%';
Power                      : '**';
NullCoalesce               : '??';
Hashtag                    : '#';
LeftShiftArithmetic        : '<<';
LessThan                   : '<';
MoreThan                   : '>' ->pushMode(HTML_TEXT_MODE);
LessThanEquals             : '<=';
GreaterThanEquals          : '>=';
Equals_                    : '==';
NotEquals                  : '!=';
IdentityEquals             : '===';
IdentityNotEquals          : '!==';
BitAnd                     : '&';
BitXOr                     : '^';
BitOr                      : '|';
And                        : '&&';
Or                         : '||';
MultiplyAssign             : '*=';
DivideAssign               : '/=';
ModulusAssign              : '%=';
PlusAssign                 : '+=';
MinusAssign                : '-=';
LeftShiftArithmeticAssign  : '<<=';
RightShiftArithmeticAssign : '>>=';
RightShiftLogicalAssign    : '>>>=';
BitAndAssign               : '&=';
BitXorAssign               : '^=';
BitOrAssign                : '|=';
PowerAssign                : '**=';
NullishCoalescingAssign    : '??=';
ARROW                      : '=>';
```

```
//ANGULARSPECIFIC
Component:'Component';
NgFor: 'NgFor' | 'ngFor' | 'ngfor';
NgIf:  'NgIf' | 'ngIf' | 'ngif';
Selector: 'selector';
TemplateUrl:'templateUrl';
Template:'template';
StyleUrls:'styleUrls';
StyleUrl: 'styleUrl';
Styles:'styles';
Imports:'imports';
Standalone:'standalone';
Get : 'get';
```

:AST

```
PairedTag {
 openTag:
 OpenTag{
 tagName: div,
 attributes: [
 QuotedAttribute{
 attributeName: style,
 attributeValue: "display: flex;"
 }]
 },
 innerTags: [
 PairedTag {
 openTag:
 OpenTag{
 tagName: div,
 attributes: [
 QuotedAttribute{
 attributeName: style,
 attributeValue: "margin-right: 20px;"
 }]
 },
 innerTags: [
 NormalHtmlText {
 text:


 },
```

codesnap.dev

```
 Class {
 GenericStatementList[
 VariableDecl {
 varName:  VariableNaming {
 VariableName: selectedProduct
 , VariableType: any
}
 , varValue: null
 },
 VariableDecl {
 varName:
 VariableNaming {
 VariableName: images
}
 , varValue:
 ArrayInfoValue {
 arrayValues: [
 JsonObjectValue{
 {src=StringValue: 'assets/Image/1.jpg', name=StringValue: 'Product 1', details=StringValue: 'Details of Product 1.'}
 }]
 }
 },
```

:Symbol Table

```
public class SymbolTable {  26 usages
    public List<Row> rowList=new ArrayList<>();  5 usages
    public void addRow(Row row) { this.rowList.add(row); }
    public void printTable(){  1 usage
        System.out.println("Type\t\t\t\t\t\t\t\t\tValue");
        System.out.println("-------------------------------------------------------------------");
        for(int i=0;i<rowList.size();i++){
            if(rowList.get(i)!=null){
                String type=rowList.get(i).type;
                String value=rowList.get(i).value;
                type=String.format("%-20s",type);
                value=String.format("%-20s",value);
                System.out.println(type +"\t\t\t" + "|" + "\t\t\t" + value);
                System.out.println("-------------------------------------------------------------------
                System.out.println();
            }
        }
    }
}
```

Type	Value
Imported	Component
Imported	CommonModule
Imported	NgFor
Imported	NgIf
VariableName	selectedProduct
VariableType	any
VariableName	images