

الجمهورية العربية السورية

جامعة دمشق

كلية الهندسة المعلوماتية



## مشروع مترجمات اللغة Angular



مشروع مادة المترجمات للسنة الرابعة – قسم هندسة البرمجيات

### إعداد الطلاب:

علياء أحمد الجاروف

سمر أحمد علوش

صفا بسام خليفة

سها علي العيسى

بثيرنه محمود السبيريج

### إشراف:

م. أية شحادة

م. وسيم بزرة

## What is ANTLR?

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

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

في سياق مترجمات اللغات البرمجية:

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

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

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

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 متعددة، سواء كانت لغات برمجة أو تنسيقات ملفات أخرى.

**تكامل مع أنظمة متعددة:** يمكن توليد محللات باستخدام ANTLR لتناسب مجموعة متعددة من الأنظمة والمشاريع.

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



## What is Angular?

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

Angular يسهل 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 خياراً شائعاً 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.

## القوالب (Templates)



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

## التوجيهات (Directives)



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

## الخدمات (Services)



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

## حقن التبعية (Dependency Injection)



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

## التوجيه (Routing)



نظام لإدارة التنقل بين صفحات التطبيق باستخدام عناوين URL.

## ربط البيانات ثنائياً الاتجاه (Two-way Data Binding)



يتيح التزامن الفوري بين واجهة المستخدم وبيانات النموذج.

## الأنباب (Pipes)



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

## خطافات دورة الحياة (Lifecycle Hooks)



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

## القابلات للملاحظة: Observables



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 اللغوي واللفظي Lexical Analysis – التحليل القواعدي Symbol – بناء شجرة Syntactical Analysis - Visitor Functionc – AST) . يتم تحقيق ذلك من خلال فهم قواعد كتابة Angular، التي تُستخدم لبناء وتنظيم واجهات المستخدم بشكل متكملاً وفعالاً.

```
function() {
  'use strict';

angular
  .module('at.account')
  .controller('AccountModalCtrl', $ctrl);

/** @ngInject */
function $ctrl($scope, $uibModalInstance, toastr) {
  $scope.panel = 0;

  $scope.setPanel = function(panelId) {
    $scope.panel = panelId;
  }

  $scope.showLogin = function() {
    $scope.setPanel(0);
  }
  $scope.showRegister = function() {
    $scope.setPanel(1);
  }

  $scope.successfulLogin = function() {
    toastr.success('Successfully logged in');
    $uibModalInstance.close();
  }

  $scope.successfulRegistration = function() {
    toastr.success('Confirmation mail sent');
    $scope.showLogin();
  }

  $scope.closeModal = function() {
    $uibModalInstance.dismiss();
  }
}
```

## التحليل اللغوي واللفظي :Lexical 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 الوحدات 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 الـ **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**:

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

2.تعريف الكلمات المحجوزة بهياكل التحكم والحلقات **Loops and Control Flow**

3.تعريف الكلمات المحجوزة الخاصة بالصفوف **Classes** والبرمجة الكائنية **OOP**:

4.تعريف أوامر الطباعة:

5.تعريف التوابع:

6.تعريف العمليات بأنواعها:

+ عمليات المقارنة:

 العمليات الرياضية:

 العمليات المنطقية:

 عمليات الإسناد والزيادة والنقصان:

## 7. تحليل السلسل النصية:

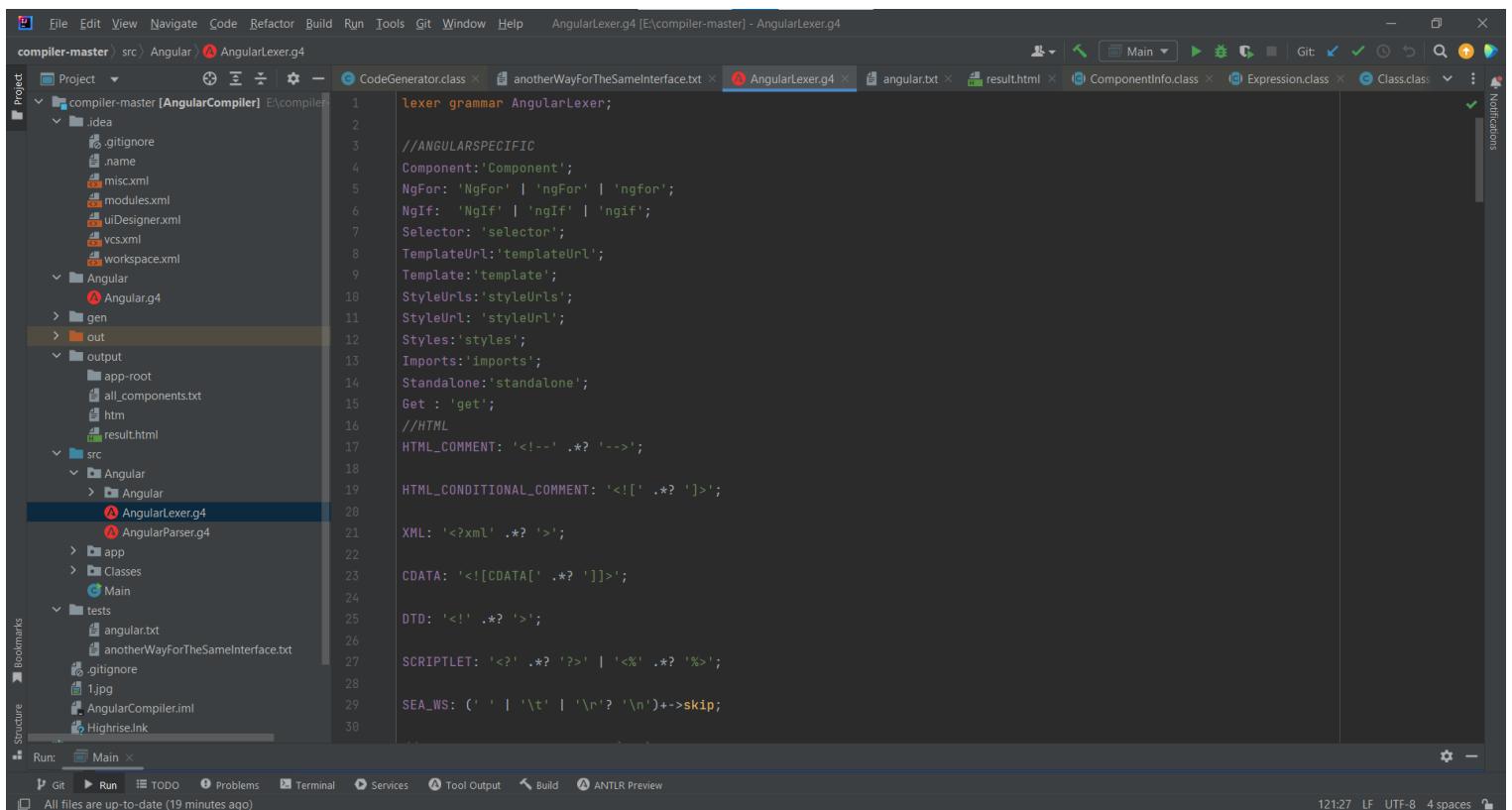
## 8.تعريف المتحولات ضمن سلسلة نصية:

## 9. التعامل مع الأعداد:

## 10.تعريف علامات الترقيم:

## 11.تعريف الأقواس بمختلف أنواعها:

## 12. المساحات البيضاء والأسطر الجديدة والتعليقات:



The screenshot shows the IntelliJ IDEA interface with the AngularCompiler project open. The code editor displays an ANTLR grammar file named AngularLexer.g4. The grammar defines various tokens such as Component, NgFor, NgIf, Selector, TemplateUrl, Template, styleUrls, StyleUrl, Styles, Imports, Standalone, Get, HTML\_COMMENT, HTML\_CONDITIONAL\_COMMENT, XML, CDATA, DTD, and SCRIPTLET. The code editor has syntax highlighting and code completion features enabled. The project structure on the left shows files like .gitignore, name, misc.xml, modules.xml, uiDesigner.xml, vcs.xml, workspace.xml, Angular.g4, and AngularLexer.g4. The bottom status bar indicates the file is up-to-date and shows the current time as 12:27.

```
lexer grammar AngularLexer;

//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';
//HTML
HTML_COMMENT: '<!-- .*? -->';
HTML_CONDITIONAL_COMMENT: '<![.*? ]>';
XML: '<?xml .*? ?>';
CDATA: '<![CDATA[.*? ]]>';
DTD: '<!.*? ?>';
SCRIPTLET: '<? .*? ?>' | '<% .*? %>';
SEA_WS: (' ' | '\t' | '\r'? '\n')+->skip;
```

The screenshot shows the IntelliJ IDEA interface with the AngularLexer.g4 file open in the main editor. The code defines various tokens for arithmetic and logical operators, assignments, and keywords. The file is part of the AngularCompiler project.

```
// RightShiftArithmetic      : '>>';  
// RightShiftLogical        : '>>>';  
LessThan                      : '<';  
MoreThan                     : '>' ->pushMode(HTML_TEXT_MODE);  
LessThanEquals                : '<=';  
GreaterThanOrEqual           : '>=';  
Equals_                       : '=';  
NotEquals                     : '!=';  
IdentityEquals                : '===';  
IdentityNotEquals             : '!==';  
BitAnd                         : '&';  
BitXOr                        : '^';  
BitOr                         : '|';  
And                            : '&&';  
Or                             : '||';  
MultiplyAssign                : '*=';  
DivideAssign                  : '/=';  
ModulusAssign                 : '%=';  
PlusAssign                    : '+=';  
MinusAssign                   : '-=';  
LeftShiftArithmeticAssign    : '<<=';  
RightShiftArithmeticAssign   : '>>=';  
RightShiftLogicalAssign       : '>>>=';  
BitAndAssign                  : '&=';  
BitXorAssign                 : '^=';  
BitOrAssign                   : '|=';  
PowerAssign                   : '**=';  
NullishCoalescingAssign     : '?=?';  
ARROW                          : '>=';  
/// Null literals
```

The screenshot shows the IntelliJ IDEA interface with the AngularLexer.g4 file open in the main editor. The code defines various tokens for arithmetic and logical operators, assignments, and keywords. The file is part of the AngularCompiler project.

```
DecimalLiteral:  
    DecimalIntegerLiteral '.' [0-9] [0-9_]* ExponentPart?  
    | '.' [0-9] [0-9_]* ExponentPart?  
    | DecimalIntegerLiteral ExponentPart?  
;  
BigDecimalIntegerLiteral : DecimalIntegerLiteral 'n';  
fragment DecimalIntegerLiteral: '0' | [1-9] [0-9_]*;  
fragment ExponentPart: [eE] [+]? [0-9_]+;  
/// 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';
```

The screenshot shows the IntelliJ IDEA interface with the AngularLexer.g4 file open in the code editor. The code editor displays the grammar rules for the AngularLexer. The project structure on the left shows the compiler-master directory containing .idea, Angular, gen, out, src, and tests. The Angular folder contains AngularLexer.g4 and AngularParser.g4. The code editor has tabs for CodeGenerator.class, anotherWayForTheSameInterface.txt, AngularLexer.g4, angular.txt, result.html, ComponentInfo.class, Expression.class, and Class.class. The status bar at the bottom indicates "All files are up-to-date (21 minutes ago)" and the time "12:27".

```
108     DecimalLiteral:
109         DecimalIntegerLiteral '.' [0-9] [0-9_]* ExponentPart?
110         | '.' [0-9] [0-9_]* ExponentPart?
111         | DecimalIntegerLiteral ExponentPart?
112     ;
113
114     BigDecimalIntegerLiteral : DecimalIntegerLiteral 'n';
115
116     fragment DecimalIntegerLiteral: '0' | [1-9] [0-9_]*;
117     fragment ExponentPart: [eE] [+ -]? [0-9_]+;
118
119     /// Keywords
120
121     Break      : 'break';
122     Do        : 'do';
123     Instanceof : 'instanceof';
124     Typeof    : 'typeof';
125     Case       : 'case';
126     Else       : 'else';
127     New        : 'new';
128     Var        : 'var';
129     Catch      : 'catch';
130     Finally    : 'finally';
131     Return    : 'return';
132     Void       : 'void';
133     Continue   : 'continue';
134     For        : 'for';
135     Switch    : 'switch';
136     While      : 'while';
137     Debugger   : 'debugger';
138     Function_  : 'function';
139     This       : 'this';
140     With       : 'with';
```

The screenshot shows the IntelliJ IDEA interface with the AngularLexer.g4 file open in the code editor. The code editor displays the grammar rules for the AngularLexer, including future reserved words. The project structure on the left shows the compiler-master directory containing .idea, Angular, gen, out, src, and tests. The Angular folder contains AngularLexer.g4 and AngularParser.g4. The code editor has tabs for CodeGenerator.class, anotherWayForTheSameInterface.txt, AngularLexer.g4, angular.txt, result.html, ComponentInfo.class, Expression.class, and Class.class. The status bar at the bottom indicates "All files are up-to-date (22 minutes ago)" and the time "17:23".

```
155     Class      : 'class';
156     Enum       : 'enum';
157     Extends    : 'extends';
158     Super      : 'super';
159     Const      : 'const';
160     Export     : 'export';
161     Import     : 'import';
162
163     /// The following tokens are also considered to be FutureReservedWords
164     /// when parsing strict mode
165
166     Implements : 'implements';
167     Let        : 'let';
168     Private    : 'private';
169     Public     : 'public';
170     Interface  : 'interface';
171     Package    : 'package';
172     Protected  : 'protected';
173     Static     : 'static';
174
175     //keywords:
176     Any        : 'any';
177     Number     : 'number';
178     Never      : 'never';
179     Boolean    : 'boolean';
180     String     : 'string';
181     Int        : 'int';
182     Unique     : 'unique';
183     Symbol     : 'symbol';
184     Undefined  : 'undefined';
```

The screenshot shows the IntelliJ IDEA interface with the AngularCompiler project open. The code editor displays the `AngularLexer.g4` file, which defines tokens for the Angular language. The file includes sections for reserved words, implements, keywords, and symbols. The code editor has multiple tabs open, including `CodeGenerator.class`, `anotherWayForTheSameInterface.txt`, `AngularLexer.g4`, `angular.txt`, `result.html`, `ComponentInfo.class`, `Expression.class`, and `Class.class`. The bottom status bar shows the file is up-to-date and provides encoding information.

```
155 Class : 'class';
156 Enum : 'enum';
157 Extends : 'extends';
158 Super : 'super';
159 Const : 'const';
160 Export : 'export';
161 Import : 'import';
162
163 // The following tokens are also considered to be FutureReservedWords
164 // when parsing strict mode
165
166 Implements : 'implements';
167 Let : 'let';
168 Private : 'private';
169 Public : 'public';
170 Interface : 'interface';
171 Package : 'package';
172 Protected : 'protected';
173 Static : 'static';
174
175 //Keywords:
176 Any : 'any';
177 Number : 'number';
178 Never : 'never';
179 Boolean : 'boolean';
180 String : 'string';
181 Int : 'int';
182 Unique : 'unique';
183 Symbol : 'symbol';
184 Undefined : 'undefined';
```

# المحلل القواعدي Parser

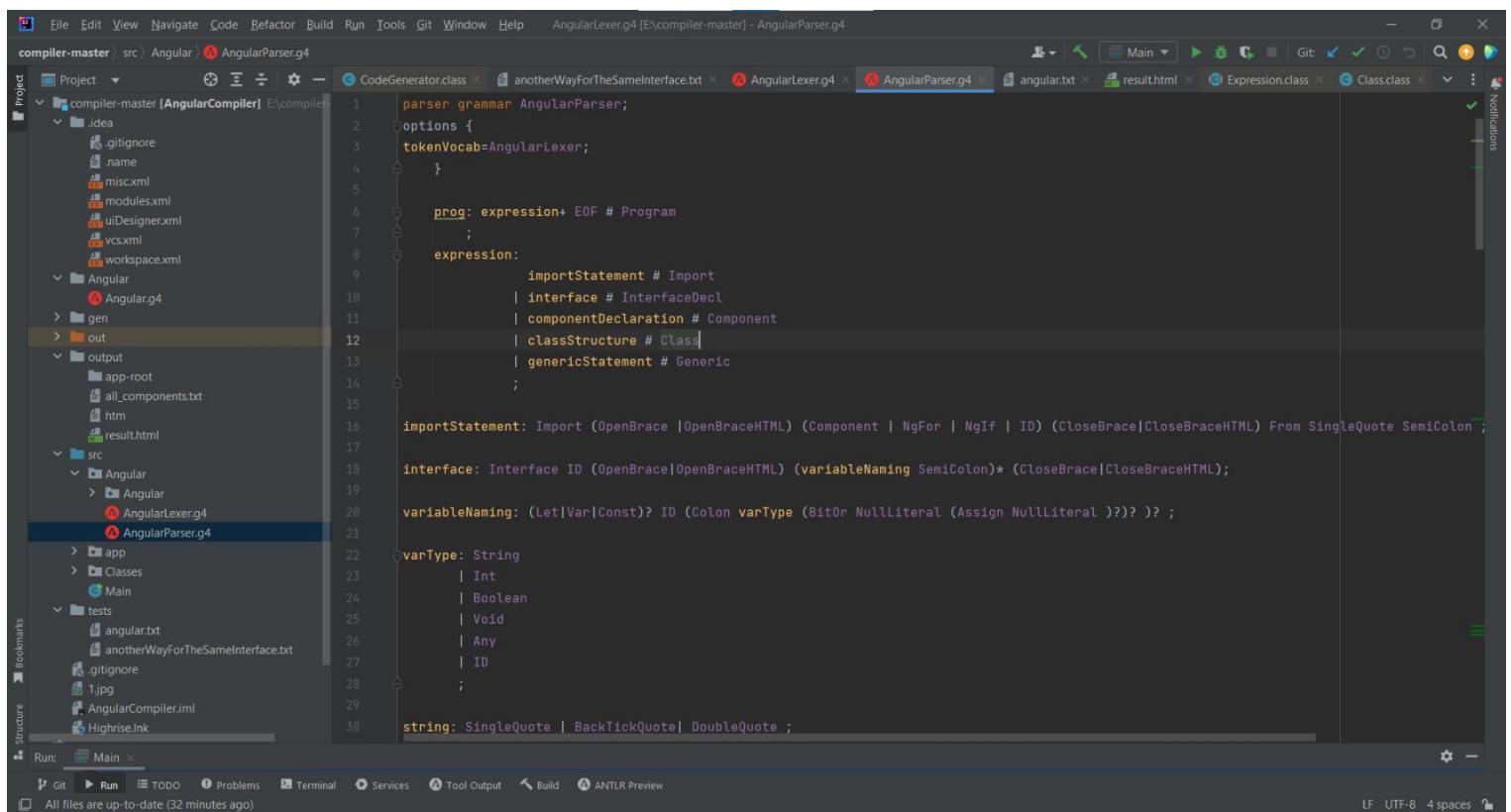
الـ **Parser** هو برنامج أو جزء من البرنامج وظيفته فهم وقراءة النصوص أو الأكواد وتحويلها إلى هيكل يمكن للكمبيوتر التعامل معه.

## . الفكرة الأساسية:

1. يستقبل الكود كمجموعة من الحروف أو الأسطر.
2. يحل النص → يفرق بين العناصر المختلفة (مثل المتغيرات، الدوال، الكلاسات).
3. يبني هيكل بيانات (مثل شجرة أو قائمة) تمثل الكود بطريقة منتظمة.
4. يقدر بعدها البرنامج يولد أكواد جديدة أو ينفذ العمليات المطلوبة بناءً على التحليل.

## . الفوائد:

- يسهل تحويل الكود البرمجي لشيء قابل للعرض أو التنفيذ بدون ما تحتاج تكتب كل شيء يدوي.
- مهم في أدوات تحليل الكود، تحويل الكود، أو بناء صفحات ديناميكية من كود موجود.



The screenshot shows an IDE interface with the AngularParser.g4 file open in the main editor. The code is a ANTLR4 grammar for parsing Angular syntax. The grammar defines a parser grammar named AngularParser with rules for tokens, expressions, statements, and specific Angular constructs like imports and interfaces. The IDE's project structure on the left shows files like AngularLexer.g4, angular.txt, and result.html. The bottom status bar indicates the code is up-to-date.

```
parser grammar AngularParser;
options {
    tokenVocab=AngularLexer;
}

prog: expression+ EOF # Program
;

expression:
    importStatement # Import
    | interface # InterfaceDecl
    | componentDeclaration # Component
    | classStructure # Class
    | genericStatement # Generic
;

importStatement: Import (OpenBrace |OpenBraceHTML) (Component | NgFor | NgIf | ID) (CloseBrace|CloseBraceHTML) From SingleQuote SemiColon ;
interface: Interface ID (OpenBrace|OpenBraceHTML) (variableNaming SemiColon)* (CloseBrace|CloseBraceHTML);
variableNaming: (Let|Var|Const)? ID (Colon varType (BitOr NullLiteral (Assign NullLiteral )?)? )? ;
varType: String
    | Int
    | Boolean
    | Void
    | Any
    | ID
;

string: SingleQuote | BackTickQuote| DoubleQuote ;
```

File Edit View Navigate Code Refactor Build Run Tools Git Window Help AngularLexer.g4 [E:\compiler-master] - AngularParser.g4

Project compiler-master [AngularCompiler] E:\compiler

genericStatement:  
    value # ValueType  
    | variableDeclaration # VariableDecl  
    | arrayDeclaration # ArrayDecl  
    | assignStatement # Assign  
    | returnStatement # Return  
    | ifStatement # If  
    | forLoop # For

;

classStructure: Export Class ID (OpenBrace|OpenBraceHTML) (genericStatement (SemiColon)?)\* (CloseBrace|CloseBraceHTML);

variableDeclaration: variableNaming (Assign value (BitOr NullLiteral (Assign NullLiteral)?))?(SemiColon)?;

arrayDeclaration: variableNaming (OpenBracket CloseBracket)? (Assign arrayInfo)\*(SemiColon);  
arrayInfo: OpenBracket value (Comma value)\* (Comma)? CloseBracket;

functionDeclaration: ID functionBody;  
functionBody: OpenParen (variableNaming (Comma variableNaming)\* CloseParen ARROW? (OpenBrace|OpenBraceHTML) genericStatement\* (CloseBrace|CloseBraceHTML));  
functionCall: ID OpenParens (value (Comma value)\* CloseParens;

assignStatement: (thisorId)? ID Assign value SemiColon;  
thisorId: ((ID|This) Dot);  
returnStatement: Return (thisorId)? value SemiColon;

ifStatement: If OpenParen conditionalState CloseParen (OpenBrace|OpenBraceHTML) genericStatement\* (CloseBrace|CloseBraceHTML) (Else genericStatement\* (CloseBrace|CloseBraceHTML));  
conditionalState: logicalStatement(logicalOp logicalStatement)\* # ConditionalStatement  
    | ID # VariableName

logicalOp: LessThanEquals  
    | GreaterThanEquals  
    | Equals\_  
    | NotEquals  
    | IdentityEquals  
    | IdentityNotEquals  
    | BitAnd  
    | BitXOr  
    | BitOr  
    | And  
    | Or

;

logicalStatement: (value (logicalOp) value);

forLoop: For OpenParen Let ID Of ID CloseParen forBody;

forBody: genericStatement # SingleLineForLoop  
    | (OpenBrace|OpenBraceHTML) genericStatement+ (CloseBrace|CloseBraceHTML) # MultipleLinesForLoop  
    | ;

jsonObject: (OpenBrace|OpenBraceHTML) ID Colon value (Comma ID Colon value)\* (CloseBrace|CloseBraceHTML);

attribute: ngForStatement # NgFor  
    | ngIfStatement # NgIf  
    | (ID|Class) Assign DoubleQuote # DoubleQuotedAttribute  
    | OpenBracket (ID|Class) CloseBracket Assign DoubleQuote # OpenBracketAttribute // [src]  
    | OpenParens (ID|Class) CloseParens Assign DoubleQuote # OpenParensAttribute // (click)  
    | ;

Run: Main

All files are up-to-date (33 minutes ago)

55:66 LF UTF-8 4 spaces

File Edit View Navigate Code Refactor Build Run Tools Git Window Help AngularLexer.g4 [E:\compiler-master] - AngularParser.g4

Project compiler-master [AngularCompiler] E:\compiler

logicalOp: LessThanEquals  
    | GreaterThanEquals  
    | Equals\_  
    | NotEquals  
    | IdentityEquals  
    | IdentityNotEquals  
    | BitAnd  
    | BitXOr  
    | BitOr  
    | And  
    | Or

;

logicalStatement: (value (logicalOp) value);

forLoop: For OpenParen Let ID Of ID CloseParen forBody;

forBody: genericStatement # SingleLineForLoop  
    | (OpenBrace|OpenBraceHTML) genericStatement+ (CloseBrace|CloseBraceHTML) # MultipleLinesForLoop  
    | ;

jsonObject: (OpenBrace|OpenBraceHTML) ID Colon value (Comma ID Colon value)\* (CloseBrace|CloseBraceHTML);

attribute: ngForStatement # NgFor  
    | ngIfStatement # NgIf  
    | (ID|Class) Assign DoubleQuote # DoubleQuotedAttribute  
    | OpenBracket (ID|Class) CloseBracket Assign DoubleQuote # OpenBracketAttribute // [src]  
    | OpenParens (ID|Class) CloseParens Assign DoubleQuote # OpenParensAttribute // (click)  
    | ;

Run: Main

All files are up-to-date (33 minutes ago)

55:66 LF UTF-8 4 spaces

The screenshot shows the IntelliJ IDEA interface with the AngularParser.g4 file open in the main editor. The code is an ANTLR grammar for parsing Angular syntax. The grammar includes rules for JSON objects, attributes, ngFor statements, ngIf statements, HTML tags, and interpolation. The code is annotated with line numbers and highlights specific tokens like ID, Colon, and DoubleQuote.

```
110 ;
111 ;
112 jsonObject:(OpenBrace|OpenBraceHTML) ID Colon value (Comma ID Colon value)* (CloseBrace|CloseBraceHTML);
113 ;
114 attribute: ngForStatement # NgFor
115 | ngIfStatement # NgIf
116 | (ID|Class) Assign DoubleQuote # DoubleQuotedAttribute
117 | OpenBracket (ID|Class) CloseBracket Assign DoubleQuote # OpenBracketAttribute // [src]
118 | OpenParen (ID|Class) CloseParen Assign DoubleQuote # OpenParenAttribute // (click)
119 ;
120 ngForStatement: Multiply NgFor Assign (Let ID Of ID)
121 ;
122 ;
123 ngIfStatement: Multiply NgIf Assign (ID | logicalStatement)
124 ;
125 ;
126 htmlTags: openTag (htmlTags)* closeTag # PairedTag
127 | selfClosingTag # UnpairedTag
128 | TEXT # NormalHtmlText
129 | interpolation # HtmlInterpolation
130 ;
131 ;
132 ;
133 interpolation: (OpenBrace|OpenBraceHTML)(OpenBrace|OpenBraceHTML) (value)* (CloseBrace|CloseBraceHTML)(CloseBrace|CloseBraceHTML);
134 openTag: (LessThan |OpenTag) ID (attribute)* (MoreThan|CloseTag);
135 closeTag: (LessThan |OpenTag) Divide ID (MoreThan|CloseTag);
136 selfClosingTag:(LessThan |OpenTag) ID (attribute)* Divide (MoreThan|CloseTag);
```

# Symbol table

ما هو Symbol Table ؟

(Interpreter) أو المفسر (Compiler) هو هيكل بيانات يستخدمه المترجم (Symbol Table) لتخزين معلومات عن الرموز أو العناصر الموجودة في البرنامج أثناء التحليل.

• الرموز تشمل:

- المتغيرات (int x, string name)
- الدوال (function sum(a, b))
- الكلاسات أو الـ Objects
- الثوابت

ما الهدف منه؟

1. تخزين معلومات الرموز مثل:

- اسم الرمز (x, y)
- نوعه (int, string)
- نطاقه → محلّي أو عام (Scope)
- مكانه في الذاكرة
- قيمته إذا كانت معروفة أثناء التحليل

2. تسهيل عمليات التحقق:

- هل هذا المتغير معرف قبل الاستخدام؟
- هل الدالة تستدعي بعد تعريفها؟
- التحقق من تطابق الأنواع بين المتغيرات والدوال.

3. تسريع البحث عن الرموز أثناء الترجمة أو التنفيذ.

File Edit View Navigate Code Refactor Build Run Tools Git Window Help AngularLexer.g4 [E:\compiler-master] - SymbolTable.java

compiler-master > src > Classes > SymbolTable > printTableWithoutErrors

Project

compiler-master [AngularCompiler] E:\compiler

- > .idea
- > Angular
- > gen
- > out
- > output
  - app-root
  - all\_components.txt
  - htm
  - result.html
- > src
  - Angular
  - app
  - Classes
    - ComponentInfos
    - GenericStatements
    - SymbolTable
      - Row
      - SymbolTable
    - Values
    - Visitor
  - CodeGenerator
  - ComponentDeclaration
  - ComponentInfo
  - Expression
  - ExpressionProcessor
  - Import
  - InterfaceBody
  - InterfaceDecl
  - LabelManager
  - Program
  - SemanticChecker

CodeGenerator.class anotherWayForTheSameInterface.txt AngularLexer.g4 Row.java SymbolTable.java AngularParser.g4 angular.txt result.html

Notifications

```
1 package Classes.SymbolTable;
2
3 import ...
4
5 public class SymbolTable {
6     7 usages
7     public List<Row> rowList=new ArrayList<>();
8
9     21 usages
10    public void addRow(Row row) { this.rowList.add(row); }
11
12    24 usages
13    public List<Row> getRows() { return this.rowList; }
14
15    1 usage
16    public void printTableWithoutErrors(List<String> errorVariables) {
17        System.out.println("\n"); //:جدول المرموز (بدون المتغيرات التي تحتوي على خطأ)
18        System.out.println("-----");
19        System.out.println("|\t|\t|\t|\t|\t|\t|\t|\t|"); //:النهاية
20        System.out.println("-----");
21
22        for(Row row : rowList) {
23            if(row != null) {
24                //تحقق مما إذا كان المتغير يحتوي على خطأ
25                boolean hasError = false;
26                for(String errorVar : errorVariables) {
27                    if(row.value.contains(errorVar)) {
28                        hasError = true;
29                        break;
30                    }
31                }
32            }
33        }
34        //طباعة الصف فقط إذا لم يكن يحتوي على خطأ
35        if(!hasError) {
36            String type = String.format("%-20s", row.type);
37            String value = String.format("%-20s", row.value);
38            System.out.printf("|\t%s\t|\t%s\n", type, value);
39            System.out.println("-----");
40        }
41    }
42
43    2 usages
44    public void printTable(){
45        System.out.println("\n"); //:جدول المرموز الكامل
46        System.out.println("-----");
47        System.out.println("|\t|\t|\t|\t|\t|\t|\t|\t|"); //:النهاية
48        System.out.println("-----");
49        for(int i=0;i<rowList.size();i++){
50            if(rowList.get(i)!=null){
51                String type=rowList.get(i).type;
52                String value=rowList.get(i).value;
53                type=String.format("%-20s",type);
54                value=String.format("%-20s",value);
55                System.out.printf("|\t%s\t|\t%s\n", type, value);
56                System.out.println("-----");
57            }
58        }
59    }
60}
```

Run: Main

Git Run TODO Problems Terminal Services Tool Output ANTLR Preview

All files are up-to-date (44 minutes ago)

18:55 LF UTF-8 4 spaces

File Edit View Navigate Code Refactor Build Run Tools Git Window Help AngularLexer.g4 [E:\compiler-master] - SymbolTable.java

compiler-master > src > Classes > SymbolTable > printTable

Project

compiler-master [AngularCompiler] E:\compiler

- > .idea
- > Angular
- > gen
- > out
- > output
  - app-root
  - all\_components.txt
  - htm
  - result.html
- > src
  - Angular
  - app
  - Classes
    - ComponentInfos
    - GenericStatements
    - SymbolTable
      - Row
      - SymbolTable
    - Values
    - Visitor
  - CodeGenerator
  - ComponentDeclaration
  - ComponentInfo
  - Expression
  - ExpressionProcessor
  - Import
  - InterfaceBody
  - InterfaceDecl
  - LabelManager
  - Program
  - SemanticChecker

CodeGenerator.class anotherWayForTheSameInterface.txt AngularLexer.g4 Row.java SymbolTable.java AngularParser.g4 angular.txt result.html

Notifications

```
33 طباعة الصف فقط إذا لم يكن يحتوي على خطأ
34 if(!hasError) {
35     String type = String.format("%-20s", row.type);
36     String value = String.format("%-20s", row.value);
37     System.out.printf("|\t%s\t|\t%s\n", type, value);
38     System.out.println("-----");
39 }
40
41
42
43 public void printTable(){
44     System.out.println("\n"); //:جدول المرموز الكامل
45     System.out.println("-----");
46     System.out.println("|\t|\t|\t|\t|\t|\t|\t|\t|"); //:النهاية
47     System.out.println("-----");
48     for(int i=0;i<rowList.size();i++){
49         if(rowList.get(i)!=null){
50             String type=rowList.get(i).type;
51             String value=rowList.get(i).value;
52             type=String.format("%-20s",type);
53             value=String.format("%-20s",value);
54             System.out.printf("|\t%s\t|\t%s\n", type, value);
55             System.out.println("-----");
56         }
57     }
58 }
```

Run: Main

Git Run TODO Problems Terminal Services Tool Output ANTLR Preview

All files are up-to-date (42 minutes ago)

LF UTF-8 4 spaces

# Semantic & Syntax Errors

في البرمجة، الأخطاء (Errors) التي تواجه المطورين تنقسم عادةً إلى فئتين رئيسيتين:

- ( .1 ) أخطاء الصياغة (Syntax Errors)
- ( .2 ) أخطاء المعنى أو المنطق (Semantic Errors)

الفرق الرئيسي هو:

- تتعلق بطريقة كتابة الكود . Syntax Errors
- تتعلق بمعنى الكود أو المنطق الذي ينفذه . Semantic Errors

## Syntax Errors

التعريف:

هي الأخطاء التي تحدث عندما تكتب الكود بطريقة غير صحيحة وفق قواعد اللغة البرمجية.

خصائصها:

- يكتشفها المترجم (Compiler) أو المفسر (Interpreter) فور محاولة تشغيل الكود.
- تمنع البرنامج من التنفيذ نهائياً حتى تصحيحها.
- 

## Semantic Errors

التعريف:

هي الأخطاء التي تحدث عندما يكون الكود صحيحاً من ناحية الصياغة، لكن البرنامج لا يفعل ما يُراد منه. بمعنى آخر، الكود "يُعمل" لكن النتائج خاطئة.

خصائصها:

- لا يكتشفها المترجم عادةً، لأن الكود مطابق للقواعد.
- تظهر عند تشغيل البرنامج أو عند نتائج خاطئة.
- غالباً ما تحتاج اختبار (Testing) أو مراجعة منطقية للكود لاكتشافها.

SemanticChecker.java

```
package Classes;
import ...;

/**
 * فاحص الالزات
 * هذه الكلاس مسؤولة عن التحقق من صحة الكود من الناحية الدلائلية
 * يقوم بفحص المتغيرات والمتغيرات وألواع البيانات
 */
3 usages
public class SemanticChecker {
    25 usages
    private SymbolTable symbolTable;
    34 usages
    private List<String> errors;
    12 usages
    private List<SymbolTable> errorSymbolTables;
    /**
     * ملخص الفاحص الالزاني
     *
     * @param symbolTable جدول الامور المراد فحصه
     */
    1 usage
    public SemanticChecker(SymbolTable symbolTable) {
        this.symbolTable = symbolTable;
        this.errors = new ArrayList<>();
        this.errorSymbolTables = new ArrayList<>();
    }
}
```

SemanticChecker.java

```
79     * @return الأخطاء المكتتبة قائمة
80     */
81     public List<String> getErrors() { return this.errors; }

84     /**
85      * طباعة الأخطاء المكتتبة بشكل مختصر
86      * يعرض لفاصيل كل خطأ مع جدول الامور المرتبط به
87      */
88     public void printErrors() {
89         if (errors.isEmpty()) {
90             System.out.println("\n" + "لا توجد أخطاء دلائلية في الكود\n");
91             return;
92         }

93         System.out.println("\n" + "=" .repeat( count: 50));
94         System.out.println("--- تفاصيل الأخطاء ---");
95         System.out.println("=".repeat( count: 50));

96         for (int i = 0; i < errors.size(); i++) {
97             String error = errors.get(i);
98             String[] parts = error.split( regex: "\n", limit: 2);
99             String errorType = parts[0];
100            String details = parts.length > 1 ? parts[1] : "";

101            System.out.println("\n" + "-" .repeat( count: 50));
102            System.out.println("رقم الخطأ: " + (i + 1));
103            System.out.println("الخطأ: " + errorType);
104            System.out.println("تفاصيل الخطأ: " + details);
105        }
106    }
}

anotherWayForTheSameInterface.txt
```

The screenshot shows an IDE interface with a Java file named `SemanticChecker.java` open. The code is annotated with Arabic comments explaining the purpose of various sections:

```
1 package Classes;
2
3 import ...
4
5 /**
6  * فل逊 الفلازت
7  * هذه الكلاس مسؤولة عن التحقق من صحة الكود من الناحية الدلالية
8  * يقوم بتحقق المتغيرات والservices وواراع البيانات
9 */
10
11 3 usages
12
13 public class SemanticChecker {
14     25 usages
15     private SymbolTable symbolTable;
16     34 usages
17     private List<String> errors;
18     12 usages
19     private List<SymbolTable> errorSymbolTables;
20
21     /**
22      * ملئني فل逊 الفلاز
23      *
24      * @param symbolTable جدول الرموز المزدوج تخصه
25      */
26
27     1 usage
28     public SemanticChecker(SymbolTable symbolTable) {
29         this.symbolTable = symbolTable;
30         this.errors = new ArrayList<>();
31         this.errorSymbolTables = new ArrayList<>();
32     }
33 }
```

The code implements a `SemanticChecker` class that takes a `SymbolTable` as a parameter. It initializes `symbolTable`, `errors` (a list of strings), and `errorSymbolTables` (a list of `SymbolTable`s).

The screenshot shows an IDE interface with a Java file named `CodeGenerator.class` open. The code is annotated with Arabic comments explaining the logic:

```
1 usage
2
3 private void checkImports() {
4     System.out.println("\n  فحص المكتوبات...\n");
5     SymbolTable errorTable = new SymbolTable();
6     boolean hasError = false;
7
8     // طباعة المكتوبات جدول الرموز للتصحيح
9     System.out.println("  المكتوبات جدول الرموز\n");
10    for (var row : symbolTable.getRows()) {
11        System.out.println("  نوع " + row.type + " : قيمة " + row.value);
12    }
13
14    // قائمة المكتوبات المطلوبة مع صيغتها الصحيحة
15    Map<String, String> requiredImports = new HashMap<>();
16    requiredImports.put("Component", "import { Component } from '@angular/core';");
17    requiredImports.put("NgModule", "import { NgModule } from '@angular/core';");
18    requiredImports.put("BrowserModule", "import { BrowserModule } from '@angular/platform-browser';");
19    requiredImports.put("FormsModule", "import { FormsModule } from '@angular/forms';");
20    requiredImports.put("HttpClientModule", "import { HttpClientModule } from '@angular/common/http';");
21    requiredImports.put("RouterModule", "import { RouterModule } from '@angular/router';");
22
23    // جمع المكتوبات الموجودة في الكود
24    List<String> foundImports = new ArrayList<>();
25    for (var row : symbolTable.getRows()) {
26        if (row.type.equals("Import") || row.type.equals("Imported")) {
27            String importText = row.value.trim();
28            if (importText.startsWith("import")) {
29                foundImports.add(importText);
30                System.out.println("  :import " + importText);
31            }
32        }
33    }
34 }
```

The code defines a `checkImports` method that prints out the current imports in the symbol table. It then creates a map of required imports and iterates through the symbol table again to find imports in the code, printing them out.

# الأخطاء

الخطأ رقم 1  
نوع الخطأ: خطأ  
التفاصيل: متغير غير معروف: 'name'  
جدول الرموز الخاطئ بهذا الخطأ:  
جدول الرموز الكامل:  
| النوع | القيمة |  
| VariableName | name  
| VariableName | image  
| VariableName | details  
| VariableName | products  
| VariableName | selectedProduct

الخطأ رقم 2  
نوع الخطأ: خطأ  
التفاصيل: متغير غير معروف: 'image'  
جدول الرموز الخاطئ بهذا الخطأ:  
جدول الرموز الكامل:  
| النوع | القيمة |

All files are up-to-date (54 minutes ago) 22:28 LF UTF-8 4 spaces

الخطأ رقم 2  
نوع الخطأ: خطأ  
التفاصيل: متغير غير معروف: 'image'  
جدول الرموز الخاطئ بهذا الخطأ:  
جدول الرموز الكامل:  
| النوع | القيمة |

الخطأ رقم 3  
نوع الخطأ: خطأ  
التفاصيل: متغير غير معروف: 'details'  
جدول الرموز الخاطئ بهذا الخطأ:  
جدول الرموز الكامل:  
| النوع | القيمة |

الخطأ رقم 4  
نوع الخطأ: خطأ  
التفاصيل: متغير غير معروف: 'products'  
جدول الرموز الخاطئ بهذا الخطأ:  
جدول الرموز الكامل:  
| النوع | القيمة |

All files are up-to-date (55 minutes ago) 22:28 LF UTF-8 4 spaces

File Edit View Navigate Code Refactor Build Run Tools Git Window Help AngularLexer.g4 [E:\compiler-master] - anotherWayForTheSameInterface.txt

compiler-master > tests > anotherWayForTheSameInterface.txt

Project Run: Main

الخطأ رقم 5  
نوع الخطأ: خطأ  
التفاصيل: متغير غير معروف: 'selectedProduct'

-----

الخطأ رقم 6  
نوع الخطأ: خطأ  
التفاصيل: لم يتم العثور على أي كومونت في الكود

-----

الخطأ رقم 7  
نوع الخطأ: خطأ  
التفاصيل: import مفقود أو غير صحيح:  
';import { RouterModule } from '@angular/router'

-----

الخطأ رقم 8  
نوع الخطأ: خطأ  
التفاصيل: import مفقود أو غير صحيح:  
';import { NgModule } from '@angular/core'

-----

الخطأ رقم 9  
نوع الخطأ: خطأ  
التفاصيل: import مفقود أو غير صحيح:  
';import { FormsModule } from '@angular/forms'

All files are up-to-date (56 minutes ago) 22:28 LF UTF-8 4 spaces

File Edit View Navigate Code Refactor Build Run Tools Git Window Help AngularLexer.g4 [E:\compiler-master] - anotherWayForTheSameInterface.txt

compiler-master > tests > anotherWayForTheSameInterface.txt

Project Run: Main

الخطأ رقم 9  
نوع الخطأ: خطأ  
التفاصيل: import مفقود أو غير صحيح:  
';import { FormsModule } from '@angular/forms'

-----

الخطأ رقم 10  
نوع الخطأ: خطأ  
التفاصيل: import مفقود أو غير صحيح:  
';import { HttpClientModule } from '@angular/common/http'

-----

الخطأ رقم 11  
نوع الخطأ: خطأ  
التفاصيل: import مفقود أو غير صحيح:  
';import { BrowserModule } from '@angular/platform-browser'

-----

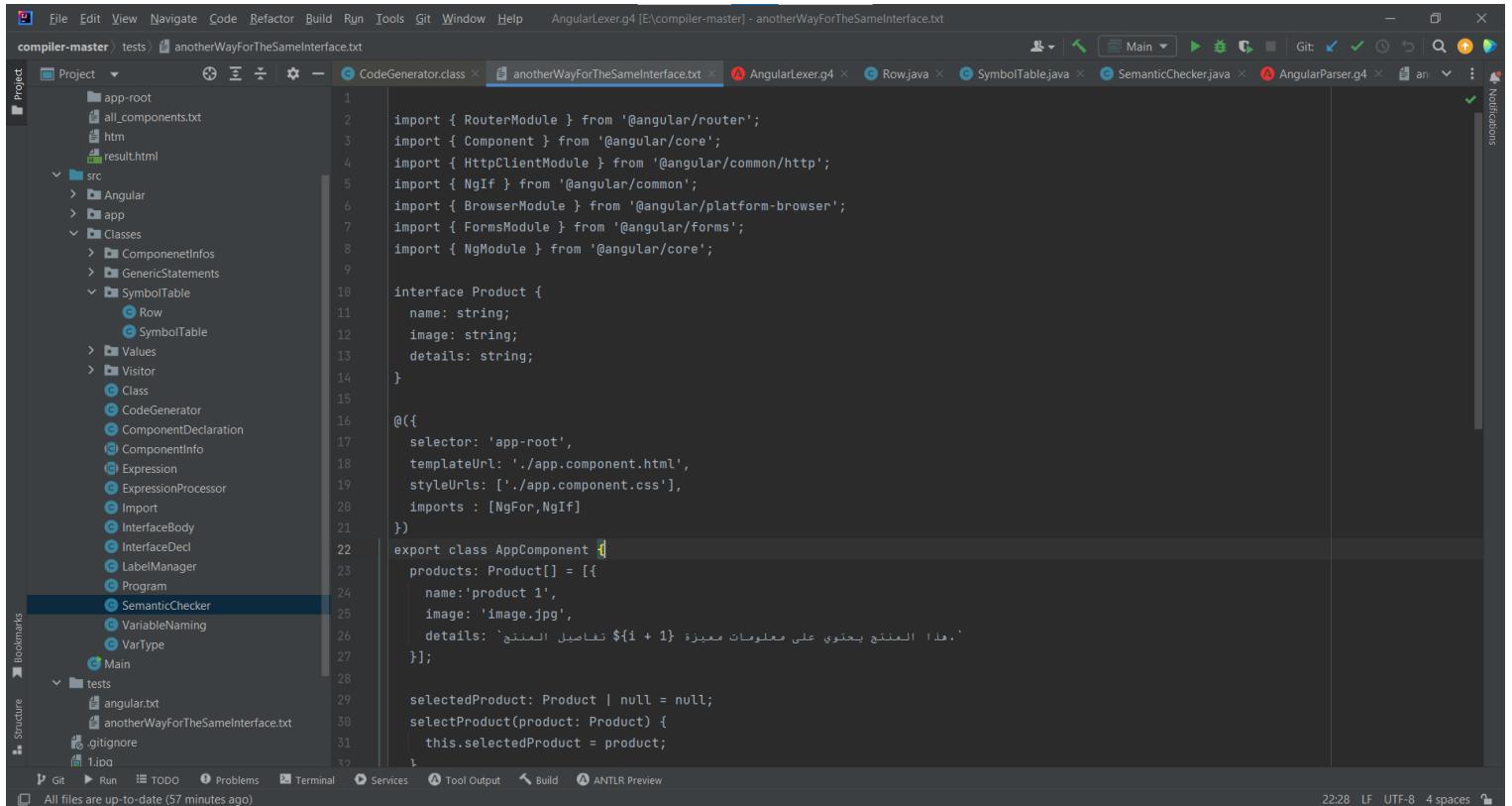
-----  
-----  
جدول الرموز (بعد حذف المتغيرات التي تحتوي على خطأ):  
-----  
-----

جدول الرموز (بدون المتغيرات التي تحتوي على خطأ):  
-----

النوع	القيمة
Import	import{Component}from'@angular/core';
Import	import{CommonModule}from'@angular/common';

All files are up-to-date (56 minutes ago) 22:28 LF UTF-8 4 spaces

# Test 1



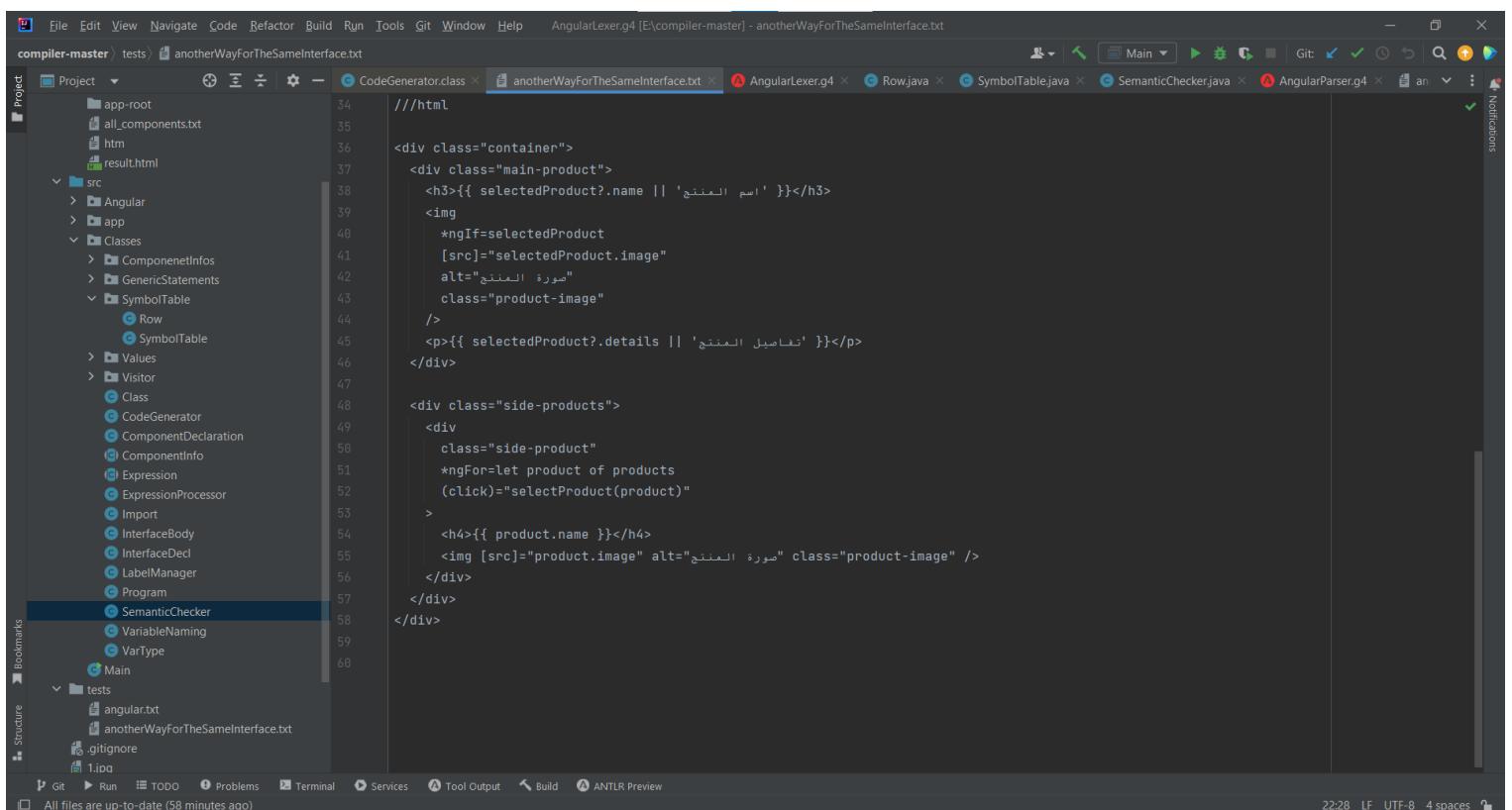
```
import { NgModule } from '@angular/router';
import { Component } from '@angular/core';
import { HttpClientModule } from '@angular/common/http';
import { NgIf } from '@angular/common';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';
import { NgModule } from '@angular/core';

interface Product {
  name: string;
  image: string;
  details: string;
}

@({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
  imports : [NgFor,NgIf]
})
export class AppComponent {
  products: Product[] = [
    {
      name:'product 1',
      image: 'image.jpg',
      details: '$1 + تفاصيل المنتج'
    }
  ];

  selectedProduct: Product | null = null;
  selectProduct(product: Product) {
    this.selectedProduct = product;
  }
}

// المنتج يحتوى على معلومات مميزة ${1 + تفاصيل المنتج}
```



```
<html>

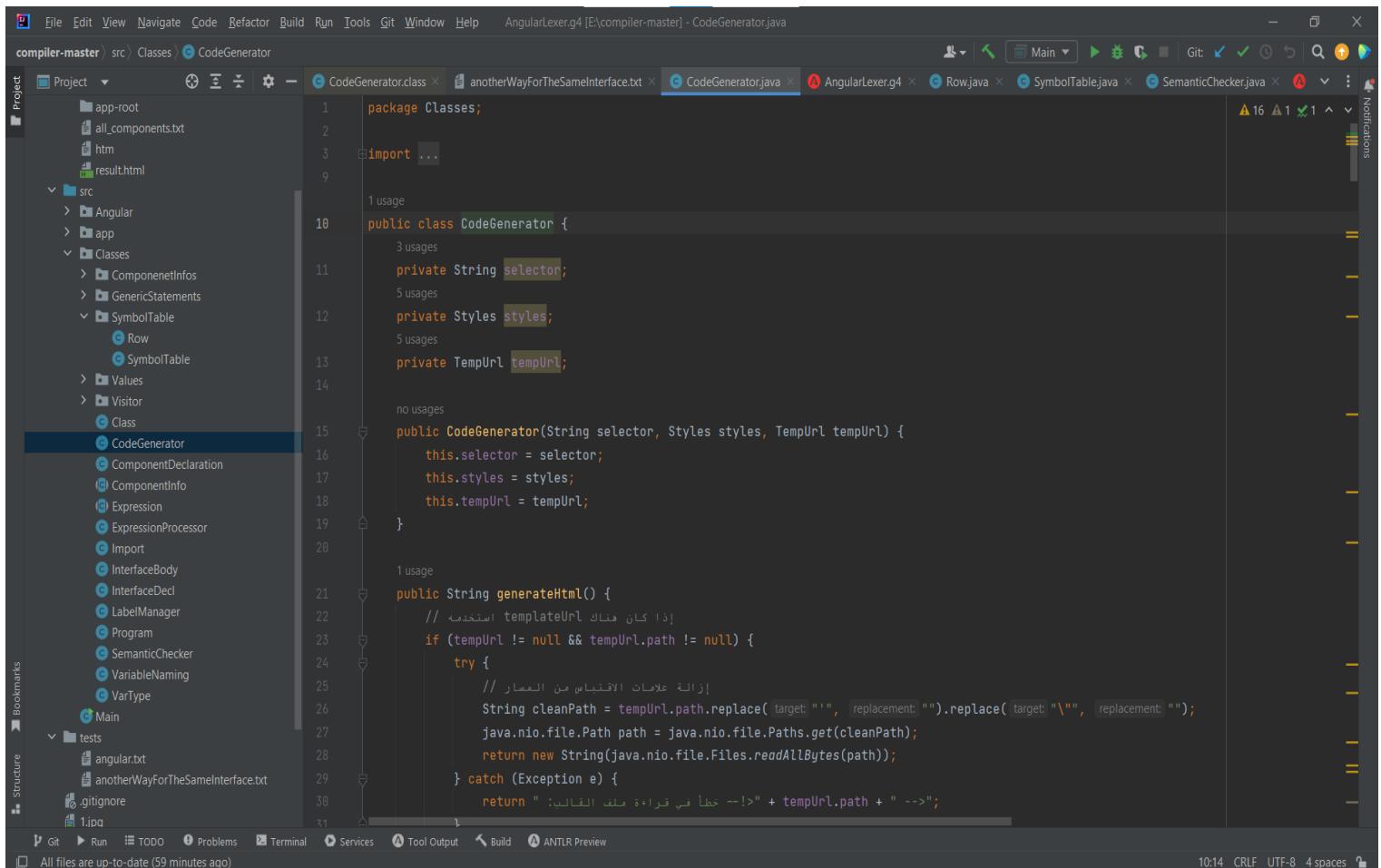
<div class="container">
  <div class="main-product">
    <h3>{{ selectedProduct?.name || 'اسم المنتج' }}</h3>
    <img
      *ngIf="selectedProduct"
      [src]="selectedProduct.image"
      alt="صورة المنتج"
      class="product-image"
    />
    <p>{{ selectedProduct?.details || 'تفاصيل المنتج' }}</p>
  </div>

  <div class="side-products">
    <div
      class="side-product"
      *ngFor="let product of products"
      (click)="selectProduct(product)"
    >
      <h4>{{ product.name }}</h4>
      <img [src]="product.image" alt="صورة المنتج" class="product-image" />
    </div>
  </div>
</div>
```

# Code generation

هو عملية تحويل نموذج أعلى مستوى من الكود أو التصميم **High-Level Source Code** إلى كود قابل للتنفيذ **Executable Code Representation** بلغة أخرى، معنى آخر: هو المرحلة التي يُنتج فيها المترجم أو أداة التطوير الكود النهائي الذي سيعمل على الحاسوب.

- جزء أساسي من **Compiler** المترجم
- يربط بين التحليل (**Analysis**) و التحسين (**Optimization**) وبين البرنامج النهائي.



The screenshot shows an IDE interface with the following details:

- File Bar:** File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, Git, Window, Help.
- Project Bar:** AngularLexer.g4 [E:\compiler-master] - CodeGenerator.java
- Toolbars:** Notifications, Main.
- Left Sidebar:** Project, Bookmarks, Structure. The Project view shows a file tree with folders like app-root, src, tests, and files like all\_components.txt, result.html, angular.txt, anotherWayForTheSameInterface.txt, etc.
- Code Editor:** The main editor window displays the Java code for the CodeGenerator class. The code defines a constructor that takes selector, styles, and tempUrl as parameters and initializes them. It also contains a generateHTML method that handles template URLs and returns a String.
- Bottom Status Bar:** All files are up-to-date (59 minutes ago), 10:14, CRLF, UTF-8, 4 spaces.

```
package Classes;
import ...;

public class CodeGenerator {
    private String selector;
    private Styles styles;
    private TempUrl tempUrl;

    public CodeGenerator(String selector, Styles styles, TempUrl tempUrl) {
        this.selector = selector;
        this.styles = styles;
        this.tempUrl = tempUrl;
    }

    public String generateHTML() {
        // إذا كان هناك templateUrl // استخدمنا
        if (tempUrl != null && tempUrl.path != null) {
            try {
                // إزالة علامات الاقتباع من المسار
                String cleanPath = tempUrl.path.replace( target: "", replacement: "" ).replace( target: "\", replacement: "" );
                java.nio.file.Path path = java.nio.file.Paths.get(cleanPath);
                return new String(java.nio.file.Files.readAllBytes(path));
            } catch (Exception e) {
                return "خطأ في قراءة ملف القالب: " + tempUrl.path + " -->";
            }
        }
    }
}
```

File Edit View Navigate Code Refactor Build Run Tools Git Window Help AngularLexer.g4 [E:\compiler-master] - CodeGenerator.java

compiler-master src Classes > CodeGenerator

Project

app-root  
all\_components.txt  
htm  
result.html

src  
> Angular  
> app  
> Classes  
> ComponenetInfos  
> GenericStatements  
> SymbolTable  
> Row  
> SymbolTable  
> Values  
> Visitor  
> Class  
CodeGenerator  
ComponentDeclaration  
ComponentInfo  
Expression  
ExpressionProcessor  
Import  
InterfaceBody  
InterfaceDecl  
LabelManager  
Program  
SemanticChecker  
VariableNaming  
VarType

Main  
tests  
angular.txt  
anotherWayForTheSameInterface.txt  
gitignore  
1.ipa

CodeGenerator.class anotherWayForTheSameInterface.txt CodeGenerator.java AngularLexer.g4 Row.java SymbolTable.java SemanticChecker.java

16 1 1 1 ^ notifications

```
public String generateCss() {
    if (styles != null && styles.paths != null && !styles.paths.isEmpty()) {
        StringBuilder sb = new StringBuilder();
        for (String style : styles.paths) {
            sb.append(style).append("\n");
        }
        return sb.toString();
    }
    return "/* No styles found */";
}

1 usage
public String generateJs() { return "// JS logic for component: " + (selector != null ? selector : "") + "\n"; }

no usages
public void writeToFile(String outputDir) throws IOException {
    java.io.File dir = new java.io.File(outputDir);
    if (!dir.exists()) dir.mkdirs();
    String html = generateHtml();
    String css = generateCss();
    String js = generateJs();

    try (FileWriter htmlWriter = new FileWriter( fileName: outputDir + "/index.html")) {
        htmlWriter.write(html);
    }
    try (FileWriter cssWriter = new FileWriter( fileName: outputDir + "/styles.css")) {
        cssWriter.write(css);
    }
    try (FileWriter jsWriter = new FileWriter( fileName: outputDir + "/main.js")) {
        jsWriter.write(js);
    }
}
```

All files are up-to-date (1 hour ago)

10:14 CRLF UTF-8 4 spaces

File Edit View Navigate Code Refactor Build Run Tools Git Window Help AngularLexer.g4 [E:\compiler-master] - CodeGenerator.java

compiler-master src Classes > CodeGenerator

Project

app-root  
all\_components.txt  
htm  
result.html

src  
> Angular  
> app  
> Classes  
> ComponenetInfos  
> GenericStatements  
> SymbolTable  
> Row  
> SymbolTable  
> Values  
> Visitor  
> Class  
CodeGenerator  
ComponentDeclaration  
ComponentInfo  
Expression  
ExpressionProcessor  
Import  
InterfaceBody  
InterfaceDecl  
LabelManager  
Program  
SemanticChecker  
VariableNaming  
VarType

Main  
tests  
angular.txt  
anotherWayForTheSameInterface.txt  
gitignore  
1.ipa

CodeGenerator.class anotherWayForTheSameInterface.txt CodeGenerator.java AngularLexer.g4 Row.java SymbolTable.java SemanticChecker.java

16 1 1 1 ^ notifications

```
no usages
public static String productsArrayToHtml(List<Map<String, String>> products) {
    StringBuilder html = new StringBuilder();
    html.append("<div class='products-container'>\n");
    for (Map<String, String> product : products) {
        html.append("  <div class='product-card'>\n");
        if (product.containsKey("image")) {
            html.append("    <img src='').append(product.get("image")).append(" alt='").append(product.getOrDefault( key: "name", defaultValue: "Product" )).append("'\n");
        }
        if (product.containsKey("name")) {
            html.append("    <h3>").append(product.get("name")).append("</h3>\n");
        }
        if (product.containsKey("details")) {
            html.append("    <p>").append(product.get("details")).append("</p>\n");
        }
        html.append("  </div>\n");
    }
    html.append("</div>\n");
    return html.toString();
}

no usages
public static String selectedProductToHtml(Map<String, String> product) {
    if (product == null) return "";
    StringBuilder html = new StringBuilder();
    html.append("<div class='product-details'>\n");
    html.append("  <h2>بيانات المنتج المختارة</h2>\n");
    if (product.containsKey("image")) {
        html.append("  <img src='').append(product.get("image")).append(" alt='").append(product.getOrDefault( key: "name", defaultValue: "Product" )).append("'\n");
    }
    if (product.containsKey("name")) {
```

All files are up-to-date (1 hour ago)

10:14 CRLF UTF-8 4 spaces

File Edit View Navigate Code Refactor Build Run Tools Git Window Help AngularLexer.g4 [E:\compiler-master] - CodeGenerator.java

compiler-master > src > Classes > CodeGenerator

Project Main Notifications

app-root  
all\_components.txt  
htm  
result.html  
src  
Angular  
app  
Classes  
ComponentInfos  
GenericStatements  
SymbolTable  
Row  
SymbolTable  
Values  
Visitor  
Class  
CodeGenerator  
ComponentDeclaration  
ComponentInfo  
Expression  
ExpressionProcessor  
Import  
InterfaceBody  
InterfaceDecl  
LabelManager  
Program  
SemanticChecker  
VariableNaming  
VarType  
Main  
tests  
angular.txt  
anotherWayForTheSameInterface.txt  
.gitignore  
Tipq

```
1 usage
public static String componentMetaToHtml(String selector, String templateUrl, String styleUrls) {
    StringBuilder html = new StringBuilder();
    html.append("<header>\n");
    if (selector != null) {
        html.append(" <div data-selector='').append(selector).append("'></div>\n");
    }
    if (templateUrl != null) {
        html.append(" <div data-template-url='').append(templateUrl).append("'></div>\n");
    }
    if (styleUrls != null) {
        html.append(" <div data-style-url='").append(styleUrls).append("'></div>\n");
    }
    html.append("</header>\n");
    return html.toString();
}

// HTML لتنزيل دوال وتحويلها إلى دوال JS وذرار
10 usages
public static class MethodInfo {
    public String name;
    5 usages
    public String params;
    1 usage
    public String body;
}

// استخراج جميع الدوال من كود
1 usage
public static List<MethodInfo> extractMethods(String angularCode) {
    List<MethodInfo> methods = new java.util.ArrayList<>();
    // { ... } methodName(params)
    java.util.regex.Matcher m = java.util.regex.Pattern.compile(
        regex "[A-Za-z0-9_]+\\$*\\(([^)]*)\\)\\$*\\$*\\{\\{([^{]*}\\}\\}\\}", java.util.regex.Pattern.DOTALL
    ).matcher(angularCode);
    while (m.find()) {
        String name = m.group(1);
        String params = m.group(2).trim();
        String body = m.group(3).trim();
        // تجاهل getters/setters أو constructors
        if (!name.equals("constructor") && !name.startsWith("get") && !name.startsWith("set")) {
            MethodInfo info = new MethodInfo();
            info.name = name;
            info.params = params;
            info.body = body;
            methods.add(info);
        }
    }
    return methods;
}

// توليد كود JS بسيط لكل دالة
1 usage
public static String generateJsForMethods(List<MethodInfo> methods) {
    StringBuilder js = new StringBuilder();
    for (MethodInfo m : methods) {
        js.append("function ").append(m.name).append("(").append(m.params).append(") {\n");
        // توليد كود بسيط: باسم الدالة و المعلمات
        js.append("    // alert('").append(m.name).append("')\n");
        js.append("    alert('").append(m.name).append("')\n");
    }
}
```

Git Run TODO Problems Terminal Services Tool Output Build ANTLR Preview

All files are up-to-date (1 hour ago) 10:14 CRLF UTF-8 4 spaces

File Edit View Navigate Code Refactor Build Run Tools Git Window Help AngularLexer.g4 [E:\compiler-master] - CodeGenerator.java

compiler-master > src > Classes > CodeGenerator

Project Main Notifications

app-root  
all\_components.txt  
htm  
result.html  
src  
Angular  
app  
Classes  
ComponentInfos  
GenericStatements  
SymbolTable  
Row  
SymbolTable  
Values  
Visitor  
Class  
CodeGenerator  
ComponentDeclaration  
ComponentInfo  
Expression  
ExpressionProcessor  
Import  
InterfaceBody  
InterfaceDecl  
LabelManager  
Program  
SemanticChecker  
VariableNaming  
VarType  
Main  
tests  
angular.txt  
anotherWayForTheSameInterface.txt  
.gitignore  
Tipq

```
1 usage
public static List<MethodInfo> extractMethods(String angularCode) {
    List<MethodInfo> methods = new java.util.ArrayList<>();
    // { ... } methodName(params)
    java.util.regex.Matcher m = java.util.regex.Pattern.compile(
        regex "[A-Za-z0-9_]+\\$*\\(([^)]*)\\)\\$*\\$*\\{\\{([^{]*}\\}\\}\\}", java.util.regex.Pattern.DOTALL
    ).matcher(angularCode);
    while (m.find()) {
        String name = m.group(1);
        String params = m.group(2).trim();
        String body = m.group(3).trim();
        // تجاهل getters/setters أو constructors
        if (!name.equals("constructor") && !name.startsWith("get") && !name.startsWith("set")) {
            MethodInfo info = new MethodInfo();
            info.name = name;
            info.params = params;
            info.body = body;
            methods.add(info);
        }
    }
    return methods;
}

// توليد كود JS بسيط لكل دالة
1 usage
public static String generateJsForMethods(List<MethodInfo> methods) {
    StringBuilder js = new StringBuilder();
    for (MethodInfo m : methods) {
        js.append("function ").append(m.name).append("(").append(m.params).append(") {\n");
        // توليد كود بسيط: باسم الدالة و المعلمات
        js.append("    // alert('").append(m.name).append("')\n");
        js.append("    alert('").append(m.name).append("')\n");
    }
}
```

Git Run TODO Problems Terminal Services Tool Output Build ANTLR Preview

All files are up-to-date (today 11:57 AM) 10:14 CRLF UTF-8 4 spaces

Screenshot of an IDE showing Java code for generating HTML and JavaScript. The code uses reflection and string manipulation to build HTML and JS strings.

```
public static String productsArrayToHtmlWithDetails(List<Map<String, String>> products) {
    StringBuilder html = new StringBuilder();
    html.append("<div class='products-container'>\n");
    for (int i = 0; i < products.size(); i++) {
        Map<String, String> product = products.get(i);
        html.append("  <div class='product-card'>\n");
        if (product.containsKey("image")) {
            html.append("    <img src='').append(product.get("image")).append(" alt='").append(product.getOrDefault(key: "name", default: "Product " + i));
        }
        if (product.containsKey("name")) {
            html.append("    <h3>").append(product.get("name")).append("</h3>\n");
        }
        if (product.containsKey("details")) {
            html.append("    <p>").append(product.get("details")).append("</p>\n");
        }
        html.append("  </div>\n");
        html.append("  <button class='show-details-btn' onclick='showProductDetails()'>").append(i).append("</button>\n");
        html.append(" </div>\n");
    }
    html.append("</div>\n");
    html.append("<div id='selected-product-details' class='product-details'>\n");
    html.append("  <h2>تفاصيل المنتج المختار</h2>\n");
    html.append("  <div id='selected-product-content'></div>\n");
    html.append("</div>\n");
    return html.toString();
}

// JavaScript لتنزيل المنتج
public static String generateProductInteractionJs(List<Map<String, String>> products) {
    StringBuilder js = new StringBuilder();
    js.append("const products = ").append(products.toString().replace(target: "=", replacement: ":"));
    js.append(";\n");
}
```

Screenshot of an IDE showing Java code for generating CSS and HTML. The code uses reflection and string manipulation to build CSS and HTML strings.

```
public static String generateProductInteractionJs(List<Map<String, String>> products) {
    StringBuilder js = new StringBuilder();
    js.append("const products = ").append(products.toString().replace(target: "=", replacement: ":"));
    js.append(";\n");
    js.append("function showProductDetails(productId) {\n");
    js.append("  const product = products[productId];\n");
    js.append("  const detailsDiv = document.getElementById('selected-product-details');\n");
    js.append("  const contentDiv = document.getElementById('selected-product-content');\n");
    js.append("  contentDiv.innerHTML = '<h3>' + product.name + '</h3><img src=' + product.image + ' alt=' + product.name + '\';\n";
    js.append("  detailsDiv.classList.add('show');\n");
    js.append("  detailsDiv.scrollIntoView({ behavior: 'smooth' });\n");
    js.append("}\n\n");
    return js.toString();
}

// CSS التوليد النهائي لبيانات التفاعل
public static String generateProductsPageHtml(String selector, String templateUrl, String styleUrls, List<Map<String, String>> products) {
    StringBuilder html = new StringBuilder();
    html.append("<!DOCTYPE html>\n<html lang='ar' dir='rtl'>\n<head>\n  <meta charset='UTF-8'>\n  <title>عرض المنتجات</title>\n<style>\n");
    html.append(extractedCss);
    if (!extractedCss.isEmpty()) {
        html.append(extractedCss.append("\n"));
    }
    html.append(generateDefaultCss());
    html.append("  </style>\n  <script>\n");
    html.append(generateProductInteractionJs(products));
    html.append(generateJsForMethods(methods));
    html.append("  </script>\n</head>\n<body>\n");
    html.append(componentMetaToHtml(selector, templateUrl, styleUrls));
    html.append("  <h2>بيانات المنتج</h2>\n");
    html.append(productsArrayToHtmlWithDetails(products));
}
```

# Test2

The screenshot shows an IDE interface with a project named 'compiler-master'. The 'angular.txt' file is open in the editor. The code defines a Product interface and an AppComponent that implements it. The AppComponent has five products, each with a name, image, and details. Arabic comments in the code translate to 'Product details 1: This product contains information about its features.' and 'Product details 4: This product contains information about its features.'

```
import { Component } from '@angular/core';
import { CommonModule } from '@angular/common';

interface Product {
  name: string;
  image: string;
  details: string;
}

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
  standalone: true,
  imports: [CommonModule]
})
export class AppComponent {
  products: Product[] = [
    {
      name: 'product 1',
      image: 'image.jpg',
      details: 'تفاصيل المنتج 1 هذا المنتج يحتوي على معلومات مميزة.'
    },
    {
      name: 'product 5',
      image: '1.jpg',
      details: 'تفاصيل المنتج 4 هذا المنتج يحتوي على معلومات مميزة.'
    }
  ];
}
```

التنفيذ

The screenshot shows a browser window displaying the output of the Angular compiler. It shows two product cards: 'product 5' and 'product 1'. Each card displays its name, image, and details. Below each card is a 'View details' button. At the bottom, there is a 'selectProduct' button and a text input field labeled 'product'.

localhost:63342/AngularLexer.g4/AngularCompiler/output/result.html?\_ijt=uhsep9s4rnqab9i417pihjpqr&\_ij\_reload=RELOAD\_ON\_SAVE

قائمة المنتجات

product 5

product 5

تفاصيل المنتج 4 هذا المنتج يحتوي على معلومات مميزة.

عرض التفاصيل

product 1

product 1

تفاصيل المنتج 1 هذا المنتج يحتوي على معلومات مميزة.

عرض التفاصيل

selectProduct

:product