BridgeCode

# Language Specification

## Type System

### Base types

* Numeric types: byte, short, int, long, float, double; no unsigned numeric types support?
* Text types: char, string
* Boolean: bool 🡪 true / false

### Composite types

Should we define a syntax for array / list definition? Similarly to what exists in perl/ruby?

E.g. [a, b, c] would be an array, {a, b, c} a tuple…

Arrays / Lists

### Variables declaration

Support for type inference; two keywords:

* “var” for mutable variables.
* “val” for immutable values.

## Structures

### Modules

Similar to C#’s static class. No need for the “static” keyword before methods (everything is static in a module).

Maybe members of a module can be used without explicitly prefixing with the module name if imported in the current source unit (similar to VB.NET modules).

### Classes

V.next?

# Annexes

## Tips

Very important, in order for the ANTLR4 custom build tool to generate base visitor and listener, the C# project must be modified by hand (from <https://github.com/tunnelvisionlabs/antlr4cs>):

* Right click the project in Solution Explorer and select Unload Project
* Right click the project in Solution Explorer and select Edit ProjectName.csproj
* Locate the XML element for the ANTLR grammar project item, which should look like the following:

<Antlr4 Include="CustomLanguage.g4">

<Generator>MSBuild:Compile</Generator>

<CustomToolNamespace>MyProject.Folder</CustomToolNamespace>

</Antlr4>

* Locate an existing XML element according to the **MSBuild Property** column in the table above, or add one if it does not already exist. For example, to generate both the parse tree listener and visitor interfaces and base classes for your parser, update the project item to resemble the following.

<Antlr4 Include="CustomLanguage.g4">

<Generator>MSBuild:Compile</Generator>

<CustomToolNamespace>MyProject.Folder</CustomToolNamespace>

<Listener>True</Listener>

<Visitor>True</Visitor>

</Antlr4>

* Save and close the project file
* Right click the project in **Solution Explorer** and select **Reload Project**.

## Links

* Create a Language Compiler for the .NET Framework:
  + <https://msdn.microsoft.com/en-us/magazine/cc136756.aspx>
* Creating a simple parser with ANTLR:
  + <http://ivanyu.me/blog/2014/09/13/creating-a-simple-parser-with-antlr/>
* Creating your own programming language with ANTLR (Blog series):
  + <http://bkiers.blogspot.fr/2011/03/creating-your-own-programming-language.html>
* ANTLR 4 Parser Helpers:
  + <https://www.nuget.org/packages/ANTLR4.ParserHelpers/1.0.4.39873>
  + <https://github.com/philiplaureano/ANTLR4.ParserHelpers>
* ANTLR (2) Tree Parsers:
  + <http://www.antlr2.org/doc/sor.html>
  + <http://web.mit.edu/dmaze/school/6.824/antlr-2.7.0/doc/sor.html>
* ANTLR 4 C# Runtime:
  + <https://github.com/sharwell/antlr4cs>

### Tree transformation

It seems that tree parsing is not available any more…

<http://stackoverflow.com/questions/14565794/antlr-4-tree-inject-rewrite-operator>

<https://theantlrguy.atlassian.net/wiki/display/~admin/2012/12/08/Tree+rewriting+in+ANTLR+v4>

<http://www.meta-environment.org/doc/books//extraction-transformation/asfsdf-by-example/asfsdf-by-example.pdf>

<https://groups.google.com/forum/#!topic/antlr-discussion/W9zGVdd6BKQ>

* <https://github.com/GRosenberg/GenPackage>
* Having just started writing my first interpreter, I'm wondering if I will be hampered by this current restriction. I started in v3 but quickly discovered v4 and moved to that. I *think*I'm progressing correctly, and hope I can figure out all the right things to do, but I do see what I think are extraneous tokens in the parse tree. I'm still experimenting with when to use Listeners vs. Visitors (currently I have listeners to define all the symbols and a visitor to actually execute the code).
* In case anyone is interested in my 2 cents: If you need an ASTs and rewrites in ANTLR4, this would be \*my\* approach:  
  - create Java classes that serve as the nodes of the AST  
  - create the AST from the parse tree by writing Java code that does that in either a listener or visitor  
  - walk that AST using a non-ANTLR, hand written visitor  
  - create a completely new AST or modify the existing one while walking the AST  
  - repeat when necessary
* Or use the **AST tree of Eclipse JDT** if the target is a Java source instead of writing that target AST. 🡪 <http://help.eclipse.org/juno/index.jsp?topic=%2Forg.eclipse.jdt.doc.isv%2Freference%2Fapi%2Forg%2Feclipse%2Fjdt%2Fcore%2Fdom%2FAST.html>

### Examples

<http://meri-stuff.blogspot.fr/2011/08/antlr-tutorial-hello-word.html>

<http://www.bearcave.com/software/antlr/antlr_examples.html>

<http://miho.github.io/antlr-4-playground/>

<https://github.com/bkiers/tiny-language-antlr4>

<http://bkiers.blogspot.nl/2011/03/creating-your-own-programming-language.html>

[**https://dexvis.wordpress.com/2012/11/22/a-tale-of-two-grammars/**](https://dexvis.wordpress.com/2012/11/22/a-tale-of-two-grammars/)

<http://www.cs.cornell.edu/courses/cs4120/2013fa/lectures/lec08-fa13.pdf>

<http://www.program-transformation.org/Transform/ANTLR>

<http://www.codetransform.com/gcc.html> (broken link)

Blog series: <http://www.alittlemadness.com/2006/07/06/antlr-by-example-part-4-tree-parsing/>

How to decorate the parse tree:

<https://github.com/sebersole/hibernate-antlr4-poc>

<http://stackoverflow.com/questions/23487619/antlr-parse-tree-modification?rq=1>

## Javascript

* Typescript: <http://www.typescriptlang.org/> + <https://github.com/Microsoft/TypeScript>
* AtScript (Angular)
  + <https://docs.google.com/document/d/11YUzC-1d0V1-Q3V0fQ7KSit97HnZoKVygDxpWzEYW0U/edit>
  + Traceur: <https://github.com/google/traceur-compiler>
  + Assert.js: <http://angular.github.io/assert/>

## Books

<http://www.amazon.com/The-Definitive-ANTLR-4-Reference/dp/1934356999/ref=dp_ob_title_bk>

<http://www.amazon.com/dp/193435645X/?tag=stackoverfl08-20>