

# Syntax Highlighting in Alan Documentation

Tristano Ajmone

Revision 1.6

September 18, 2018

---

## Table of Contents

.....	iv
1. Syntaxes Preview .....	1
1.1. Alan .....	1
1.2. BNF .....	1
1.3. Game Transcripts .....	1
2. The Alan Syntax .....	3
2.1. Syntax Tests .....	3
2.1.1. Predefined Classes and Instances .....	3
2.1.2. Quoted Identifiers .....	4
2.1.3. Strings .....	5
2.1.4. False Positive Keywords Matches .....	5
2.1.5. Ligatures Support .....	5
3. BNF Syntax .....	7

---

# List of Figures

2.1. Example of correct ligatures support in Sublime Text. .... 6

---

This document is for previewing how source code and other verbatim blocks will be rendered and styled in the final documentation, and for comparing quality and differences across all the supported output formats.

Some contents are format dependent and will vary according to the document's output format and the backend employed, providing additional information specific to that format and/or backend.

Currently, we're still working on implementing syntax highlighting of Alan source code in the various output formats, as different conversion backends will employ different highlighting tools.

---

# Chapter 1. Syntaxes Preview

## 1.1. Alan

```
-----
-- "Barracks Example" by Tristano Ajmone
-----

The canteen IsA LOCATION
  Name 'The Barrack''s Mess Hall'.
END The canteen.

The trolley IsA object At canteen
  Container Taking object.
  Limits
    Count 5 Else "The trolley is full!"
  Has Not been_examined.
  VERB examine
  DOES
    If This Has Not been_examined
      Then "This old trolley has surely seen better days. $nIt seems barely
        capable of carrying its own weight without collapsing, let alone
        endure its daily use!"
      Make This been_examined.
      Else "It's just an old trolley."
    End If.
  END VERB examine.
End The trolley.

-----

Start at canteen.
```

## 1.2. BNF

```
class = 'EVERY' id
      [inheritance]
      {property}
      'END' 'EVERY' [id] ['.']
```

## 1.3. Game Transcripts

Although a game transcript is neither source code nor a verbatim block, I'm including it in this document to allow comparing its styles and colors to other block-type elements in the documentation.

West of House.

Welcome to Dungeon (ALAN Demo). This version created 29-FEB-92.

You are in an open field west of a big white house with a boarded front door. There is a small mailbox here.

> *examine the mailbox*

I see nothing special about the mailbox.

> *open it*

The small mailbox is now open. The small mailbox contains a leaflet.

---

## Chapter 2. The Alan Syntax

Different conversion toolchains use different tools for syntax highlighting. For HTML format conversion, various highlighters are natively supported by Asciidoctor. For PDF conversion, the usable highlighting tool(s) will be dictated by the third party backend employed.

In all cases, a custom Alan syntax definition would have to be created for the highlighter tool used with each backend, as none of them ships with a native Alan definition (the sole exception being [Highlight<sup>1</sup>](#), which is not natively supported in any of the backends, but could be integrated into some of them).

### 2.1. Syntax Tests

Here follow some code snippets for testing both common- and edge-cases in the language and ensure they are rendered as expected.

#### 2.1.1. Predefined Classes and Instances

Although it's usually unnecessary to highlight with separate styling the predefined Alan classes and instances (i.e. the `hero`), it might be desirable to be able to do so in tutorials and documentation, for educational purposes. For this reason, I've added a special highlighting group for the predefined classes, and another one for the `hero`. Both can be optionally enabled by assigning styles to them, or just be ignored (or set to the same style as normal text) to hide them in the highlighted source.

```
Synonyms me = hero.

The basement IsA location.
  Description "What a dark ang gloomy place!"
End The.

The vampire IsA actor At basement.
End The.

The coffin IsA object At basement.
  Container taking thing.
End The.

ADD TO EVERY thing
```

---

<sup>1</sup> <http://www.andre-simon.de/>

```
VERB examine
  Does
    "You examine $+1, but find nothing unusual."
  END VERB.
END ADD TO.

ADD TO EVERY string
  VERB 'say'
    Does
      "You say ""$$" Say This. "$$""!"
    END VERB.
  END ADD TO.

ADD TO EVERY integer
  VERB shout
    Does
      "You shout ""$$" Say This. "$$" out loud!"
    END VERB.
  END ADD TO.

ADD TO EVERY literal
  VERB whisper
    Does
      "You whisper ""$$" Say This. "$$""!"
    END VERB.
  END ADD TO.

-----

Start at basement.
```

## 2.1.2. Quoted Identifiers

Although quoted identifiers shouldn't receive any special coloring in the highlighted code, the syntax definition must be made aware of them to prevent false-positive keywords matches for tokens inside quoted identifiers:

```
The 'At the Bus Stop' IsA location --> Watch out for: 'AT', 'THE' and 'STOP'!
End The.
```

This hidden feature will be implemented differently in each syntax definition, depending on how the specific syntax highlighting engine works.

However, in the final highlighted code, quoted identifiers are shown with the same color and style as the base (normal) text, effectively hiding to the end user the difference between quoted identifiers and plain code (normal identifiers are just treated as plain code).



## Single Quotes Escaping

To prevent breaking up syntax highlighting, quoted identifiers need to support correctly escaping single quotes inside them via `' '`. Although this won't be visible in the highlighted output of the following example, examining the document's source (with formats which support it, like HTML) would allow to check that the quoted identifier was correctly parsed and tagged by the highlighter:

```
The 'Bob''s House' IsA location --> Escaped apostrophe in location name.
End The.
```

### 2.1.3. Strings

#### 2.1.4. False Positive Keywords Matches

The following snippet tests against false positive keywords in strings and quoted identifiers. This should never happen, and if it does then the Alan syntax definition for that highlighter is flawed and needs to be fixed.

```
-- Tokens in strings and quoted IDs shouldn't be highlighted as keywords...

The 'At the Bus Stop' IsA location --> Watch out for: 'AT', 'THE' and 'STOP'!
  Description
    "An old man waits for the bus here." --> 'AN', 'FOR', 'THE', 'HERE'!
End The.
```

#### 2.1.5. Ligatures Support

The following snippets is meant to test if the current backend and/or font supports ligatures. Some code fonts support ligatures replacements for common programming symbols, but they might differ in the way they represent `>=`, `<=`, `=>` and `=<`, depending on which combinations are assigned as comparison operators and which as arrows.

We must ensure that, if ligatures are supported in any converted document, Alan's `>=` (greater or equal), `<=` (less or equal) and `=>` (alternative `Then` in Rules) are correctly represented.

Here's the test snippet:

```
--> Test "string identity operator" ligature ('='+'=' -> '==')
```

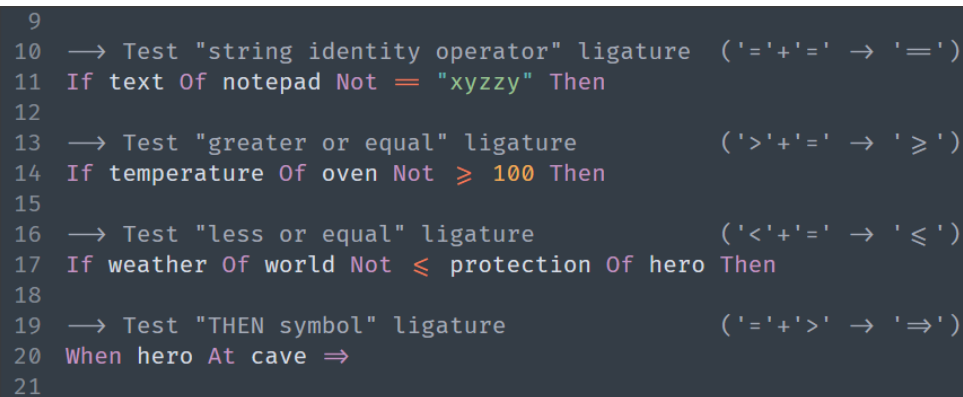
```
If text Of notepad Not == "xyzyy" Then

--> Test "greater or equal" ligature      ('>'+=' -> '>=')
If temperature Of oven Not >= 100 Then

--> Test "less or equal" ligature         ('<'+=' -> '<=')
If weather Of world Not <= protection Of hero Then

--> Test "THEN symbol" ligature          ('='+>' -> '=>')
When hero At cave =>
```

And here's a screenshot of the above code displayed in Sublime Text 3 via the [Sublime Alan<sup>2</sup>](https://github.com/tajmone/sublime-alan) package:



```
9
10 → Test "string identity operator" ligature ('='+=' → '==')
11 If text Of notepad Not = "xyzyy" Then
12
13 → Test "greater or equal" ligature      ('>'+=' → '≥')
14 If temperature Of oven Not ≥ 100 Then
15
16 → Test "less or equal" ligature         ('<'+=' → '≤')
17 If weather Of world Not ≤ protection Of hero Then
18
19 → Test "THEN symbol" ligature          ('='+>' → '⇒')
20 When hero At cave ⇒
21
```

**Figure 2.1. Example of correct ligatures support in Sublime Text.**

---

<sup>2</sup> <https://github.com/tajmone/sublime-alan>

---

## Chapter 3. BNF Syntax

Currently, BNF blocks are not being syntax highlighted but implementing a syntax definition would be very easy.

```
attribute_declaration = id
                        | 'NOT' id
                        | id integer
                        | id string
                        | id id
                        | id '{' values '}'
```

```
forms = indefinite | definite | negative

definite = 'DEFINITE' article_or_form

indefinite = [ 'INDEFINITE' ] article_or_form

negative = 'NEGATIVE' article_or_form

article_or_form = 'ARTICLE' {statement}
                 | 'FORM' {statement}
```