Turtle

-Terse RDF Triple Language

https://www.w3.org/TR/turtle/

Terse RDF Triple Language (Turtle)

- Terse RDF Triple Language
  - RDF written in a compact and natural text form
  - Plain text syntax: easy to scribble, easy to read based on Unicode
  - Mechanisms for namespace abbreviation
  - Allows grouping of triples according to subject
  - Shortcuts for collections
  - In short:
    - Takes good things of RDF/XML - an XML based format, hard to read/write
    - and leaves out angle (<>) brackets

Turtle Syntax: Default Namespaces

Syntax

```
@prefix abbr: <URI> .
```

Example:

```
@prefix uol:<http://www.cs.le.ac.uk/rdf#> .
```

Allows you to write the phrase

```
uol:Leicester
```

and it will be interpreted as

```
<http://www.cs.le.ac.uk/rdf#Leicester>
```

`uol:Leicester` means concatenate `www.cs.le.ac.uk/rdf#` with `Leicester` to give `http://www.cs.le.ac.uk/rdf#Leicester`

Turtle Syntax: Default Namespaces

You can also create a default prefix: Syntax:

```
@prefix :<URI> .
```

Example:

```
@prefix :<http://www.cs.le.ac.uk/rdf#> .
```

Allows you to write the phrase

```
:Leicester
```

and it will be interpreted as

```
<http://www.cs.le.ac.uk/rdf#Leicester>
```

Turtle Syntax: Literals

Syntax:

Literals are literal datatypes, i.e., strings, ints, etc. Literals should be enclosed in quotes but may be typed with a ^^ operator. Literals are usually typed by XSD datatypes. Two untyped literals may be represented as

```
"abc"
"123"
```

and two typed literals may be represented as

```
"abc"^^xsd:string
"123"^^xsd:integer
```

Turtle Syntax: Triples

Simple triples are a sequence of (subject, predicate, object) terms, separated by whitespace and terminated by '.' after each triple.

```
<subject> <predicate> <object> .
```

For example:

```
<http://www.cs.le.ac.uk/rdf#Leicester> <http://www.cs.le.ac.uk/
rdf#postcode>  "LE"^^xsd:string.
```

Can be written as:

```
@prefix : <http://www.cs.le.ac.uk/rdf#> .
:Leicester :postcode "LE".
```

Turtle Syntax: Groups of Triples

Same subject example:

```
@prefix ex: <http://www.example.com/> .
ex:thing ex:relation "Some text".
ex:thing ex:otherrelation ex:otherthing .
```

Can be written as:

```
@prefix ex: <http://www.example.com/> .
ex:thing ex:relation “Some text” ;
         ex:otherrelation ex:otherthing .
```

Semicolon separates statements that differ in property and value

Turtle Syntax: Groups of Triples

Same property example:

```
@prefix ex: <http://www.example.com/> .
ex:thing ex:relation "Some text".
ex:thing ex:relation ex:something .
```

Can be written as:

```
@prefix ex: <http://www.example.com/> .
ex:thing ex:relation "Some text" ,
                        ex:something .
```

Blank Nodes in Turtle

RDF blank nodes in Turtle are expressed as _: followed by a blank node label which is a series of name characters.

A fresh RDF blank node is allocated for each unique blank node label in a document (e.g. _:alice is an unique blank node label)

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

_:alice foaf:knows _:bob .
_:bob foaf:knows _:alice .
```

Nesting Unlabeled Blank Nodes in Turtle

Blank node can be represented as [ ]

John knows someone (represented as a blank node).

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
foaf:john foaf:knows [].
```

Someone knows someone else, who has the name "Bob".

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
[] foaf:knows [ foaf:name "Bob" ].
```

Blank nodes as as subject of multiple triples

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
[ foaf:name "Bob";
  foaf:mbox "<bob@exmaple.com>"].
```

Nesting Unlabeled Blank Nodes in Turtle

Example:

Using nested blank nodes:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

[ foaf:name "Alice" ] foaf:knows [
    foaf:name "Bob" ;
    foaf:knows [
        foaf:name "Eve" ] ;
     foaf:mbox <bob@example.com> ] .
```

Corresponding simple triples:

```
_:a <http://xmlns.com/foaf/0.1/name>"Alice" .
_:a <http://xmlns.com/foaf/0.1/knows> _:b .
_:b <http://xmlns.com/foaf/0.1/name> "Bob" .
_:b <http://xmlns.com/foaf/0.1/knows> _:c .
_:c <http://xmlns.com/foaf/0.1/name> "Eve" .
_:b <http://xmlns.com/foaf/0.1/mbox> <bob@example.com> .
```

Types

There's a special relationship between a thing and a category of things, called type (`http://www.w3.org/1999/02/22-rdf-syntax-ns#type`). If we want to say that Anne is a person, we can write it like this:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdf: <http: //www.w3.org/1999/02/22-rdf-syntax-ns#>.

foaf:john rdf:type foaf:Person .
```

Because this is such a fundamental relationship, Turtle has a special keyword "a", to replace the type relationship:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

foaf:john a foaf:Person .
```

Longer Example

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix ex: <http://example.org/stuff/1.0/> .

<http://www.w3.org/TR/rdf-syntax-grammar>
  dc:title "RDF/XML Syntax Specification (Revised)" ;
  dc:editor [
    ex:fullname "Dave Beckett";
    ex:homePage <http://purl.org/net/dajobe/>
  ] .
```