Ada-TOML:
a TOML parser for Ada

Pierre-Marie de Rodat, AdaCore
FOSDEM 2020 (Ada Developer room)
XML

- More or less easy to read/write
- Expensive to process, hard to get it right (namespaces, validation, ...)

```xml
<map xmlns="foo">
  <key foo:kind="string">K1</key>
  <value>V1</value>
  Extra text
</map>
```
JSON

- Simple to read/write
- Still inconvenient for humans:
  - forbidden trailing commas
  - quotes in all mappings
  - no provision for comments
- Gotchas: no requirements for number precision and no difference between floats and ints in the spec

```json
{
    "this": "JSON",
    "document": "is",
    "invalid": "!",
}
```
YAML

- Superficially simple to read/write
- Actually notoriously hard to parse correctly: the spec is very complex
- Gotchas: none is a string, null is the null value

# Are these strings equivalent?
string1: |
  Hello, world!
string2: >
  Hello, world!
string3: >-
  Hello, world!
# This is a TOML document.
title = "TOML Example"

[owner]
name = "Tom Preston-Werner"
dob = 1979-05-27T07:32:00-08:00 # First class dates

[database]
server = "192.168.1.1"
ports = [ 8001, 8001, 8002 ]
connection_max = 5000
enabled = true

https://github.com/toml-lang/toml
TOML out there

- File format for language package managers:
  - Cargo (Rust)
  - PIP, Pipenv, Poetry (Python, see PEP 518)
  - dep (Go)
  - Alire (Ada)
- Configuration file for various projects
- Implementations in lots of programming languages: C, C++, C#, Java, Go, Haskell, Java, JavaScript, Python, Ruby, Rust, ...
Ada-TOML

- Pure Ada 2012 library (3-Clause BSD License), available in Alire
- Two jobs:
  - parse bytes (TOML document) to in-memory data structures (load)
  - turn in-memory data structures into bytes (dump)
- Data structures and primitives to build/inspect in-memory data structures (much like containers)
- Subprograms to load and to dump

https://github.com/pmderodat/ada-toml
Data structures (1/3)

- In the TOML package
- TOML_Value: polymorphic value
- Any_Value_Kind: nature of the value behind a TOML_Value object

```pascal
type Any_Value_Type is
  (TOML_Table, -- Key/value mapping
   TOML_Array, -- Sequence of values
   TOML_String,
   TOML_Integer, ...);

type TOML_Value is private;
function Kind
  (Value : TOML_Value) return Any_Kind_Value;
```
Data structures (2/3)

- **TOML_Value constructors:**
  - function Create_Boolean (Value : Boolean) return TOML_Value
  - Create_Integer
  - Create_Table
  - ...

- **Getters:**
  - function As_Boolean (Value : TOML_Value) return Boolean
  - As_Integer
  - Table.Set (Key, Entry_Value)

- Tables and arrays have APIs similar to Ada.Containers
Data structures (3/3)

I : constant TOML_Value := Create_Integer (42);
S : constant TOML_Value := Create_String ("hello, world!");
T : constant TOML_Value := Create_Table;

T.Set ("int", I);
T.Set ("str", S);

-- By now, T is equivalent to the following JSON document:
--  {"int": 42, "str": "hello, world!"}

pragma Assert (T.Get ("int").As_Integer = 42);
Load/Dump (1/3)

- Still in the TOML package
- Load from in-memory strings
- Dump to in-memory string

-- Read_Result contains either an error message or a TOML_Value.

function Load_String
    (Content : String) return Read_Result

function Dump_As_String
    (Value : TOML_Value) return String
Load/Dump (2/3)

- In the TOML.File_IO package
- Load from the file system
- Dump to a Ada.Text_IO.File_Type object

```ada
function Load_File
  (Filename : String) return Read_Result

procedure Dump_To_File
  (Value : TOML_Value;
   File  : in out Ada.Text_IO.File_Type)
```
Load/Dump (3/3)

- You can make the parser/dumper work on any other stream of bytes
- TOML.Generic_Parse
- TOML.Generic_Dump

generic
  type Input_Stream (<> ) is limited private;
  with procedure Get
    (Stream : in out Input_Stream;
     EOF    : out Boolean;
     Byte   : out Character) is <>;
  Tab_Stop : Positive := 8;
function TOML.Generic_Parse
  (Stream : in out Input_Stream)
  return TOML.Read_Result
Thank you!

- [https://github.com/pmderodat/ada-toml](https://github.com/pmderodat/ada-toml)
- `ada-toml` crate in Alire
- Only one release for now: 0.1, stable enough for production use
- Contributions welcome!