RUHR-UNIVERSITÄT BOCHUM

Bridging the Gap: Secure and lossless conversion of XML data structures to the JSON format

Bachelor thesis ■ March 30, 2017 – June 29, 2017

June 20, 2017

Advisors: Dennis Felsch & Paul Rösler

Jan Holthuis

Lehrstuhl für Netz- und Datensicherheit

Overview

Last time we learned that…

- Based on my current conversion critera, *no existing solution* allows lossless XML$\rightarrow$JSON$\rightarrow$XML roundtrips…
- … but **JsonML** comes close
- I fixed the issues to allow truly lossless conversion
- Bugs in DOM implementations (`xmldom`!) can be a problem

Overview

Progress made in the last two weeks

`xmldom` breakage

- Last time I had to fix > quoting in text nodes
- Found some more bugs in `xmldom`:
  - XML declarations were treated as Processing Instructions
  - `DOCTYPE` declarations with an internal subsets broke parsing
  - Trailing whitespace in the data field of Processing Instructions was stripped

Progress made in the last two weeks

`xmldom` breakage

- Last time I had to fix > qu[illegible] t[illegible]xt nodes
- Found some mo[illegible] `x[illegible]m:`
  - XML decla[illegible]s were treated as Proces[illegible]tructions
  - `DOCTY`[illegible]larations wi[illegible]ternal subset[illegible]roke parsing
  - Trailing whitespace[illegible] th[illegible]eld of [illegible]rocessing Instructions was stripped

Progress made in the last two weeks

- Continued writing everything down (almost done!)
- Fixed some minor issues in test documents

Overview

Type Inference

JSON's implicit types

```
{
    "string": "somevalue",
    "number": 1337,
    "bool": true,
    "nothing": null
}
```

- Some converters try to be smart and guess them based on value

Type Inference

Let'try to guess types

- `hello world`

Type Inference

Let'try to guess types

- `hello world` $\rightarrow$ String

Type Inference

Let'try to guess types

- `hello world` $\rightarrow$ String
- `123`

Type Inference
Let'try to guess types

RUB

- `hello world` $\rightarrow$ String
- `123` $\rightarrow$ Number

hg NDS

Type Inference

Let’try to guess types

- `hello world` $\rightarrow$ String
- `123` $\rightarrow$ Number
- `1e-4`

Type Inference

Let'try to guess types

- `hello world` $\rightarrow$ String
- `123` $\rightarrow$ Number
- `1e-4` $\rightarrow$ Number ($= 1 \cdot 10^{-4} = 0.0001$)

Type Inference

Let'try to guess types

- `hello world` $\rightarrow$ String
- `123` $\rightarrow$ Number
- `1e-4` $\rightarrow$ Number $(= 1 \cdot 10^{-4} = 0.0001)$
- `1e-324`

Type Inference

Let'try to guess types

- `hello world` $\rightarrow$ String
- `123` $\rightarrow$ Number
- `1e-4` $\rightarrow$ Number ($= 1 \cdot 10^{-4} = 0.0001$)
- `1e-324` $\rightarrow$ Number ($= 1 \cdot 10^{-324} = 0.00 \ldots 001$)
    - But JavaScript's `parseFloat("1e-324")` will return `0`!

Type Inference

Let'try to guess types

- `hello world` $\rightarrow$ String
- `123` $\rightarrow$ Number
- `1e-4` $\rightarrow$ Number $(= 1 \cdot 10^{-4} = 0.0001)$
- `1e-324` $\rightarrow$ Number $(= 1 \cdot 10^{-324} = 0.00\ldots001)$
    - But JavaScript's `parseFloat("1e-324")` will return `0`!
- What about `1e-4` vs. `1E-4`?

Type Inference

Let'try to guess types

- `hello world` $\rightarrow$ String
- `123` $\rightarrow$ Number
- `1e-4` $\rightarrow$ Number ($= 1 \cdot 10^{-4} = 0.0001$)
- `1e-324` $\rightarrow$ Number ($= 1 \cdot 10^{-324} = 0.00 \ldots 001$)
    - But JavaScript's `parseFloat("1e-324")` will return `0`!
- What about `1e-4` vs. `1E-4`?
- If `true` is a Boolean, what about `True`?

Type Inference

Don't try to be smart!

- Inferring a type just complicates things and is error-prone
- Application can't be sure which type it'll get
  - String field incidentally contains just digits $\rightarrow$ Number type
- Just use the String type instead

Overview

Next Steps

- Finish writing and submit thesis
- Start making slides for the final presentation on July 4
- Keep my fingers crossed! 😉

Thanks!

Questions?

Reach out via email:

- **Jan Holthuis**
  jan.holthuis@rub.de