

Week 4 - Syntax

Aljaž Medič



Plan for today

1. Questions about Exercise Sheet 3?
2. Quiz
3. Syntax exercises

Questions about Exercise Sheet 3

Any questions?

(You can also drop by during the break)

QUIZ

Generic name for denormalized data is "semi-structured data". Name two markup formats that support it.

XML and JSON

List the four main building blocks of XML.

elements, attributes, text, comments

List all basic components of JSON.

strings, numbers, booleans, null, arrays, objects

QUIZ

What does it mean for a document to be 'well-formed'?

It follows the syntactic rules of the chosen language (XML/JSON),
so that parsers can read it without error.

Which syntax (XML or JSON) is better for hierarchical data in enterprise systems?

XML; because it supports deeply nested, tagged structures with mixed content.

What can't we represent in CSV (natively), that we can in JSON/XML?

More complex, nested data.

QUIZ

Both JSON and XML require well-formed syntax to be processed correctly.

True. (“You have to be more careful when writing JSON/XML than when reading it.”)

XML tends to be less verbose than JSON, thus having smaller file sizes.

False. (XML tends to be more verbose than JSON, leading to larger file sizes.)

Keys should not be repeated in JSON objects.

True.

QUIZ

JSON is human readable format, and is widely used with Web APIs

True.

JSON strictly enforces data types, meaning a number must always be interpreted as an integer or floating-point.

False. (Interpretation of numbers in JSON is left to the consumer.)

In JSON, all whitespace (except the one inside strings) is ignored by default

True.

QUIZ

In JSON, you can escape newline by using latin1 encoding and 4 hex values: "\L000a"

False. (In JSON standard, we can use unicode encoding: "\u000a")

A string in JSON can be single-quoted or double-quoted.

False. (Only double quotes are allowed for strings in JSON.)

In JSON, backslashes are used to escape special characters like quotes and newlines.

True. (escaped quote: "\""; synonyms: "\n" and "\u000a")

QUIZ

XML tag can have at most 65536 (2^{16}) attributes.

False. (The amount of attributes in the standard is not bound.)

Why do we use namespaces for in XML?

To avoid naming conflicts when different XML elements or attributes have the same name but different meaning.

If we use a default namespace in XML, all attributes are also part of it.

False. (Attributes are not part of the default namespace unless explicitly bound.)

QUIZ

Key names in JSON can start with numbers.

True. (In JS, you can access them by using `obj["1pass"]`, rather than `obj.1pass`.)

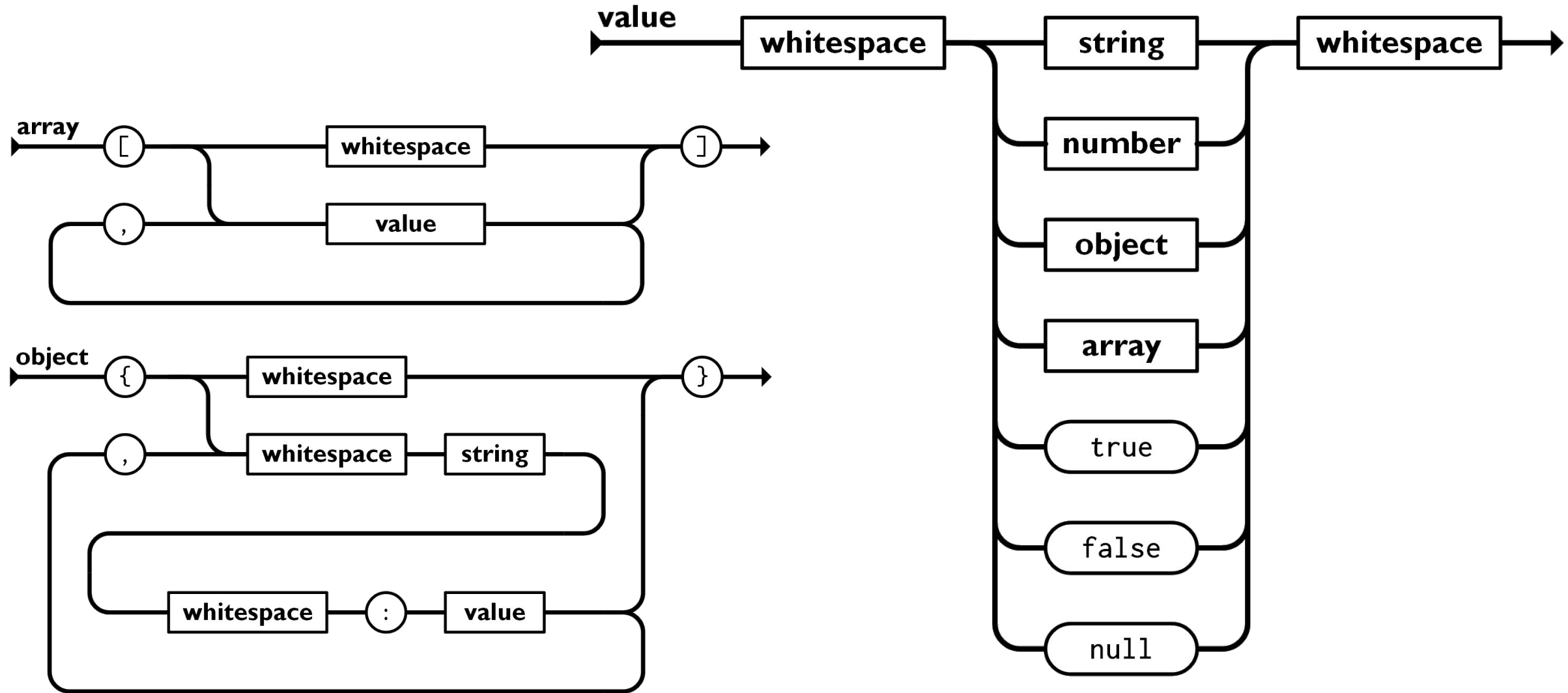
Element names in XML can start with numbers.

False. (Element names must start with a letter or underscore and cannot start with 'xml' in any case.)

In XML text, we must escape closing tag, optionally also opening tag.

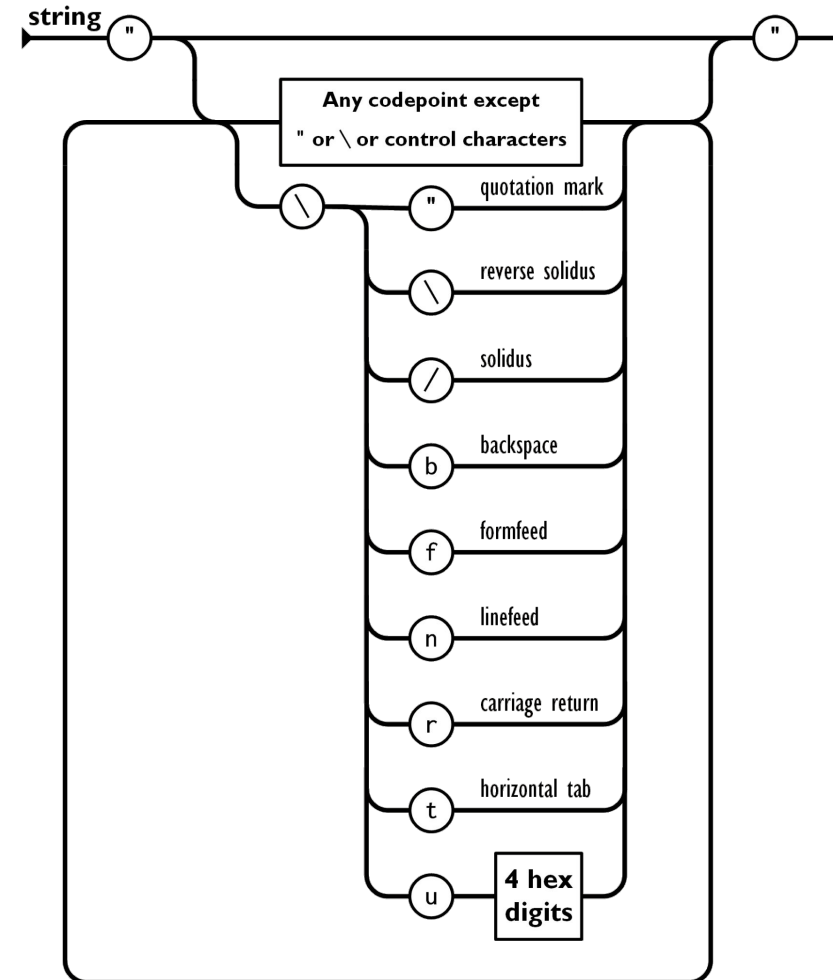
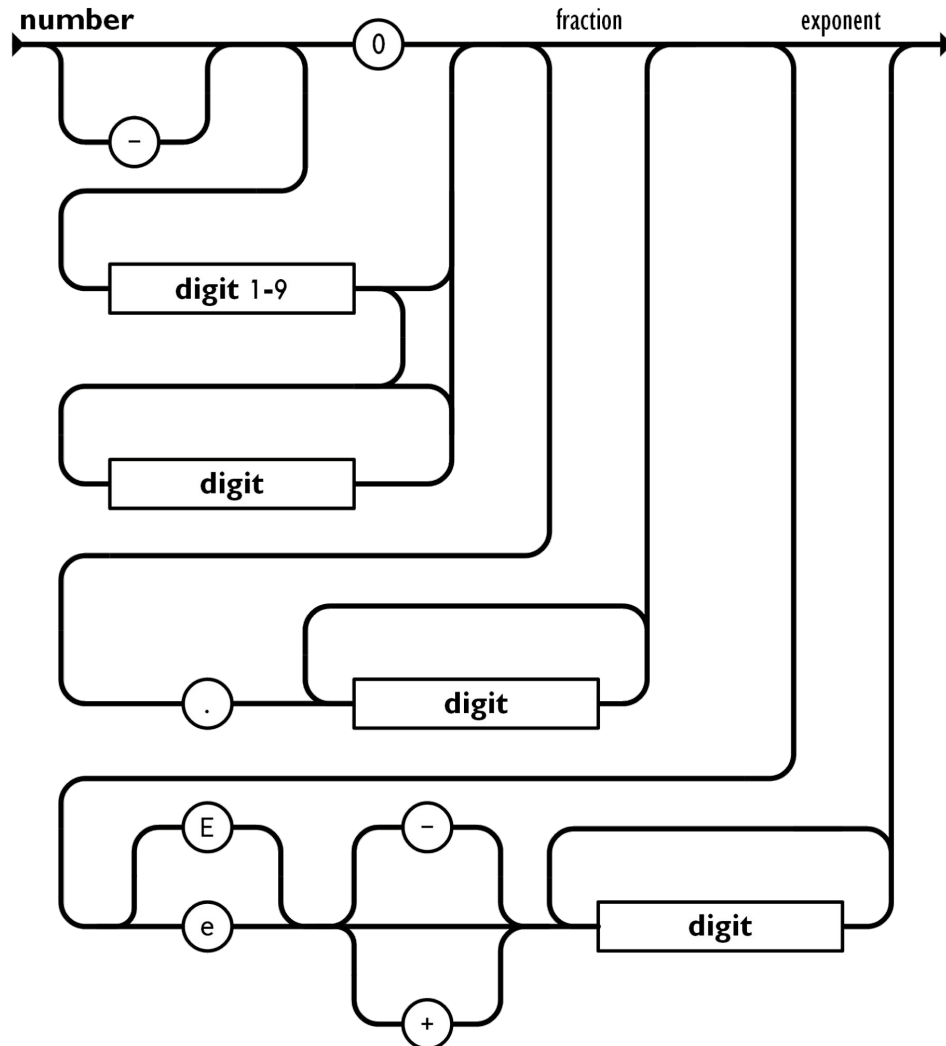
False. (It's the other way around. The reason is, so we don't confuse the parser where the new tag starts.)

JSON Document Structure



Source: json.org

JSON Document Structure (cont'd)



Source: json.org

JSON Well-formedness

```
{type:"ID", "person":["last":'Einstein', "first":'Albert'],  
"liked-wearing-socks": False, "born":{"name": "Ulm", "coordinates":[48.391750,  
9.991305]}}, "age": +76}
```

```
1 {  
2   "type": "ID",  
3   "person": {  
4     "last": "Einstein",  
5     "first": "Albert"  
6   },  
7   "liked-wearing-socks": false,  
8   "born": {  
9     "name": "Ulm",  
10    "coordinates": [  
11      48.39175,  
12      9.991305  
13    ]  
14  },  
15  "age": 76  
16 }
```

XML Document Structure

	Top-Level	Between Element Tags	Inside Opening Element Tag
Elements	once	yes	no
Attributes	no	no	yes
Text	no	yes	no

Source: Big Data Book, Ch. 5.4

XML Element Naming

Control characters															
Control characters															
SP	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

Allowed anywhere in name

Allowed but not at start

not allowed

Source: Big Data Book, Ch. 5.4

XML Well-formedness (Tags)

```
<i-love-xml/>
```

Well-formed.

```
<i-<3-xml/>
```

Not well-formed. (Element names cannot contain special characters except for `_ . - :`)

```
<i-love-xml  
>
```

Well-formed.

```
<i-&lt;3-xml/>
```

Not well-formed. (Same as 2nd)

```
<1-love-xml/>
```

Not well-formed. (Element names must start with a letter or an underscore)

```
<_i-love-xml/>
```

Well-formed.

```
<xml-i-love/>
```

Not well-formed. (Element names cannot start with xml in any case)

```
<i love xml/>
```

Well-formed. (But not what you think! It's an `<i>` tag with 2 value-less attributes)

XML Well-formedness (Documents)

```
<tree color="yellow" type='lemon' />
```

Well-formed.

```
<orchard>  
  <tree color="yellow" type='lemon' />  
  <tree type="apple" />  
</orchard>
```

Well-formed.

```
<tree type="apple" />  
<tree color="yellow" type='lemon' />
```

Not well-formed.

(Only one element on top level)

```
<Let's><Do><it/>  
</Do></Let's>
```

Not well-formed. (Tag name contains invalid characters)

```
<Mary likes="Josh" likes="Bob" />
```

Not well-formed. (Duplicate attribute name)

```
<b:city xmlns:b="http://buildings.org/xmlns">  
  <b:house b:type="terraced house"/>  
  <b:church/>  
  <b:hospital/>  
</b:city>
```

Well-formed.

ETH zürich

See you next week!

Aljaž Medič
amedic@ethz.ch



Slides



Suggestions