ETH zürich

Ghislain Fourny

Big Data for Engineers Spring 2020

7. Data Models

pinkyone / 123RF Stock Photo

Syntax vs. Data Models

Physical view

Syntax

This is syntax

```
ID,Last name,First name,Theory,
1,Einstein,Albert,"General, Special Relativity"
2,Gödel,Kurt,"""Incompleteness"" Theorem"
```

Syntax vs. Data Models

Logical view
Data Model

ID	Last name	First name	Theory
1	Einstein	Albert	General, Special Relativity
2	Gödel	Kurt	"Incompleteness" Theorem

This is a data model

Physical view
Syntax

```
ID,Last name,First name,Theory,
1,Einstein,Albert,"General, Special Relativity"
2,Gödel,Kurt,"""Incompleteness"" Theorem"
```

JSON Data Model

JSON Values

Strings

Numbers

Booleans

Null

Objects

Arrays

Atomic values

Structured values

JSON Values

Strings

Numbers

Booleans

Null

Objects
String-to-Value map

Arrays
List of values

Atomic values

Structured values

JSON Values

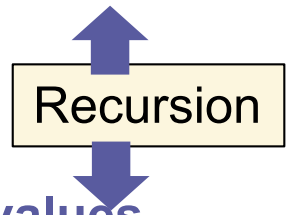
Strings

Numbers

Booleans

Null

Objects
String-to-Value map



Arrays
List of values

Atomic values

Structured values

Tree-based visual model

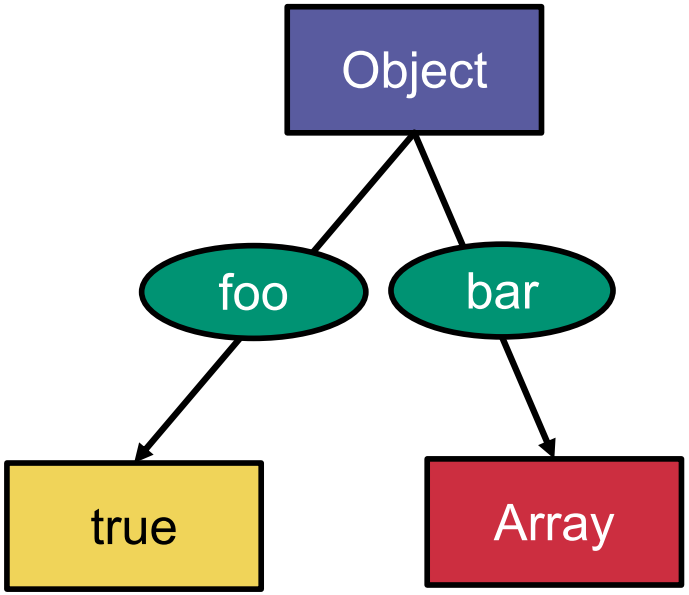
```
{
  "foo" : true,
  "bar" : [
    {
      "foobar" : "foo"
    },
    null
  ]
}
```

Tree-based visual model


```
{
  "foo" : true,
  "bar" : [
    {
      "foobar" : "foo"
    },
    null
  ]
}
```

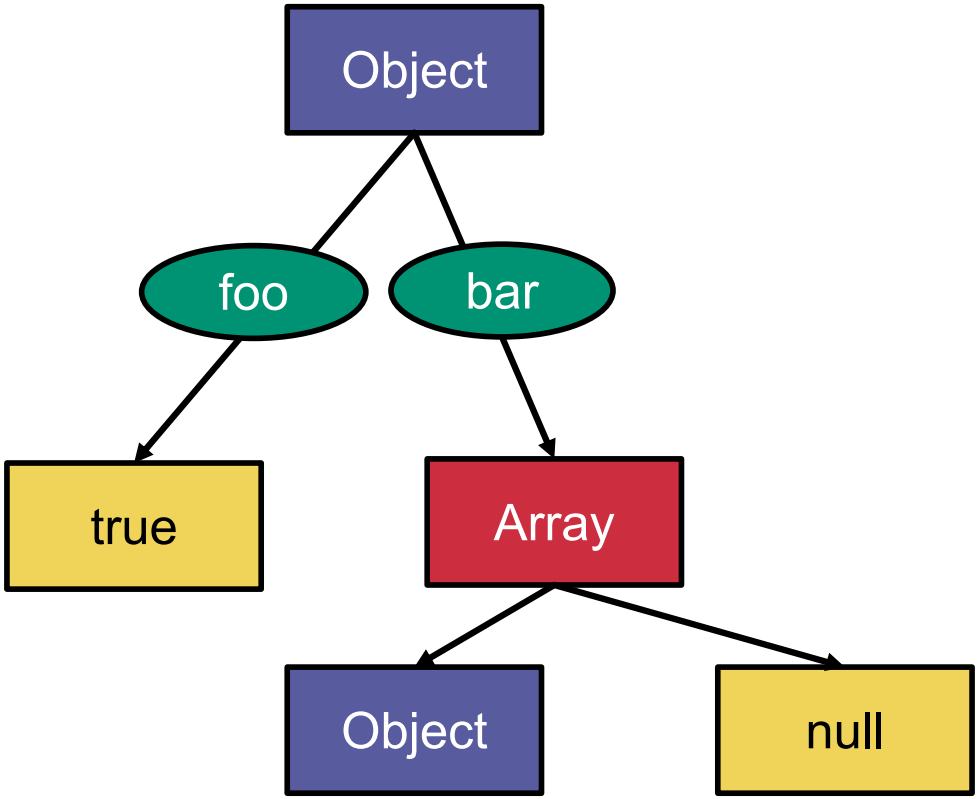
Tree-based visual model

```
{
  "foo" : true,
  "bar" : [
    {
      "foobar" : "foo"
    },
    null
  ]
}
```



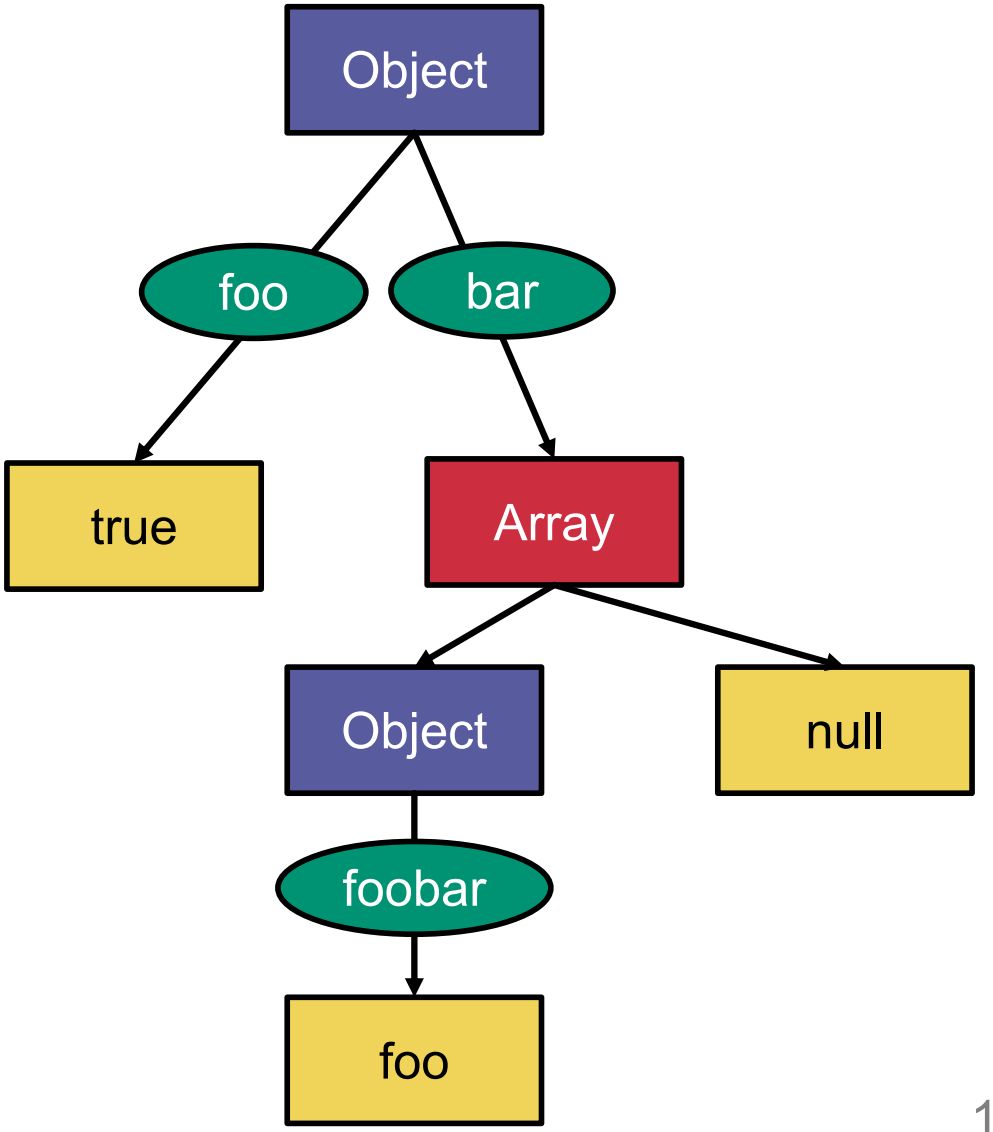
Tree-based visual model

```
{
  "foo" : true,
  "bar" : [
    {
      "foobar" : "foo"
    },
    null
  ]
}
```



Tree-based visual model

```
{
  "foo" : true,
  "bar" : [
    {
      "foobar" : "foo"
    },
    null
  ]
}
```



Syntax vs. Data Models

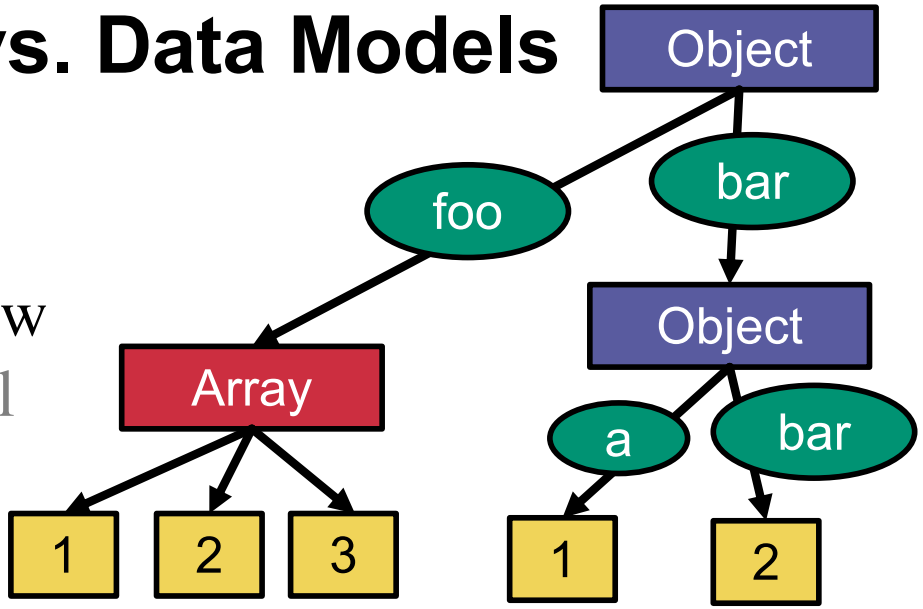
Physical view

Syntax

```
{
  "foo" : [ 1, 2, 3 ],
  "bar" : { "a" : 1, "bar" : 2 }
}
```

Syntax vs. Data Models

Logical view
Data Model



Physical view
Syntax

```
{
  "foo" : [ 1, 2, 3 ],
  "bar" : { "a" : 1, "bar" : 2 }
}
```

Syntax vs. Data Models

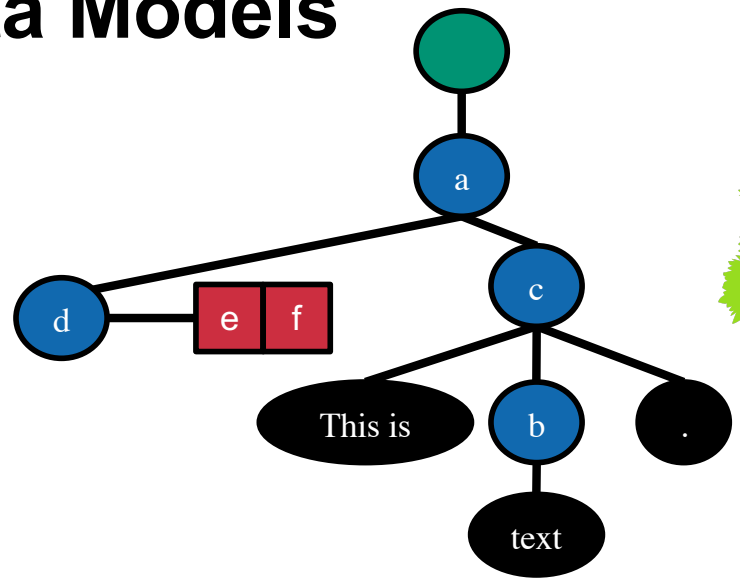
Physical view

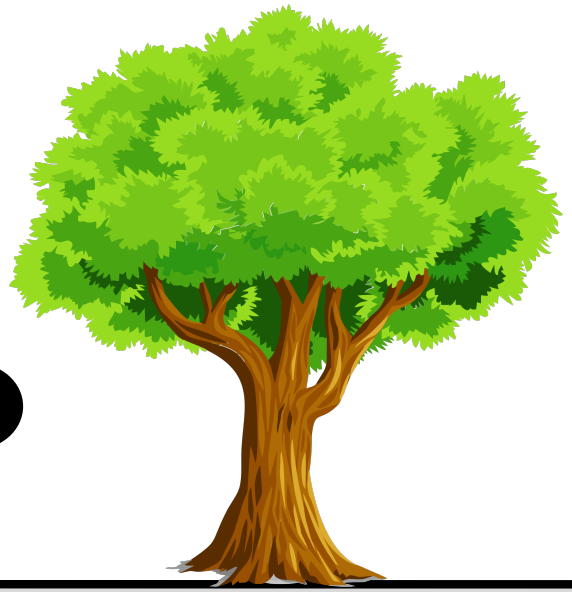
Syntax

```
<a>
  <d e="f"/>
  <c>This is <b>text</b>.</c>
</a>
```

Syntax vs. Data Models

Logical view
Data Model

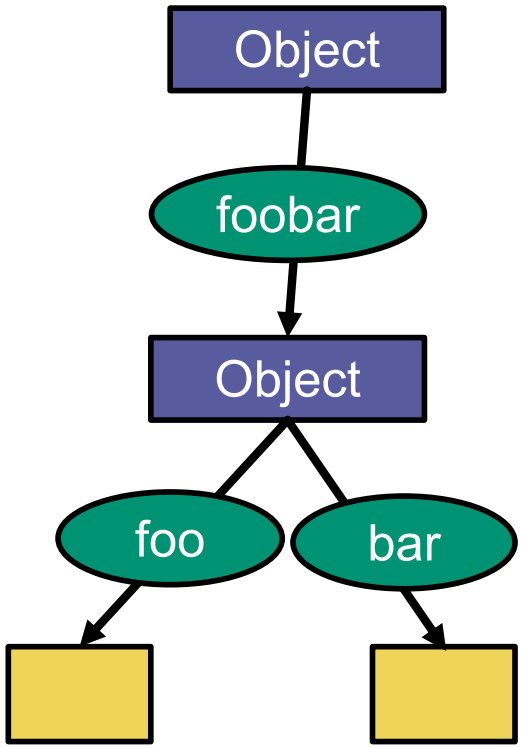




Physical view
Syntax

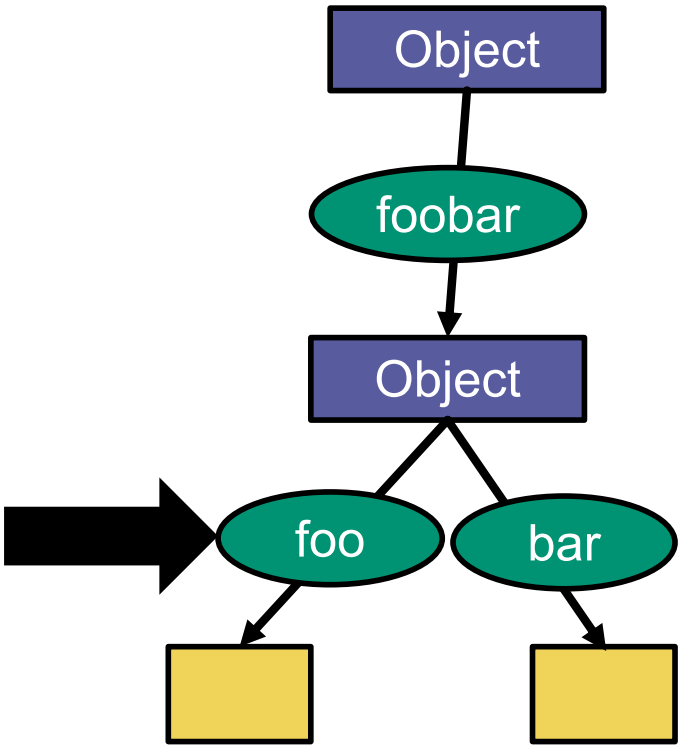
```
<a>
  <d e="f"/>
  <c>This is <b>text</b>.</c>
</a>
```

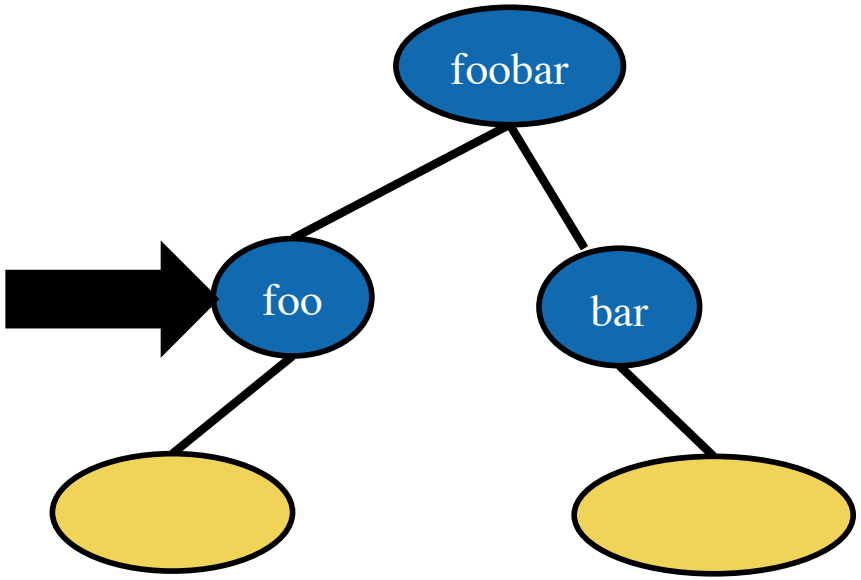
Edge vs. Node labeling



Labels are on the **edges**

Edge vs. Node labeling





JSON

Labels are on the **edges**

XML

Labels are on the **nodes**

JSON Data Models

"original" (implicit) JSON Data Model

http://www.json.org/

JSON Schema Data Model

https://www.ietf.org/archive/id/draft-wright-json-schema-01.txt

JSONiq Data Model (JDM)

http://www.jsoniq.org/docs/JSONiqExtensionToXQuery/html/section-jsoniq-data-model.html

XML Data models

Information Set (Infoset)

http://www.w3.org/TR/xml-infoset/

Post Schema-Validation Infoset (PSVI)

http://www.w3.org/TR/xmlschema11-1/

XQuery and XPath Data Model (XDM)

http://www.w3.org/TR/xpath-datamodel/

XML Information Set

Information Set

```
<?xml version="1.0" encoding="UTF-8"?>
<dc:metadata xmlns:dc="http://www.systems.ethz.ch">
    <dc:title
        language="en"
        year="2019"
        >Systems Group</dc:title>
    <dc:publisher>ETH Zurich</dc:publisher>
</dc:metadata>
```

Information Set

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE metadata>
<metadata>
    <title
        language="en"
        year="2019"
        >Systems Group</title>
    <publisher>ETH Zurich</publisher>
</metadata>
```

The 11 XML Information Items

Document
Element
Attribute
Processing Instruction
Character
Comment
Namespace
Unexpanded Entity Reference
DTD
Unparsed Entity
Notation

24

The 4 XML (most important) Information Items we cover

DocumentNEW
Element
Attribute
Processing Instruction
Character (Text)
Comment

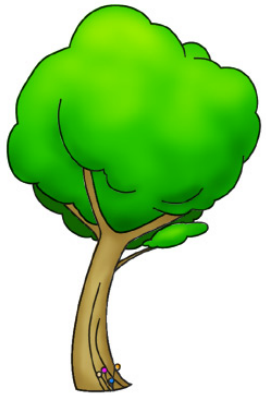
Namespace
Unexpanded Entity Reference
DTD
Unparsed Entity
Notation

25

Document Information Items

Document Information Item doc
[children] Element Information Item metadata
[version] 1.0

The Document Information Item is always present (even if the optional DOCTYPE is missing)



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE metadata>
<metadata>
    <title
        language="en"
        year="2019"
        >Systems Group</title>
    <publisher>ETH Zurich</publisher>
</metadata>
```

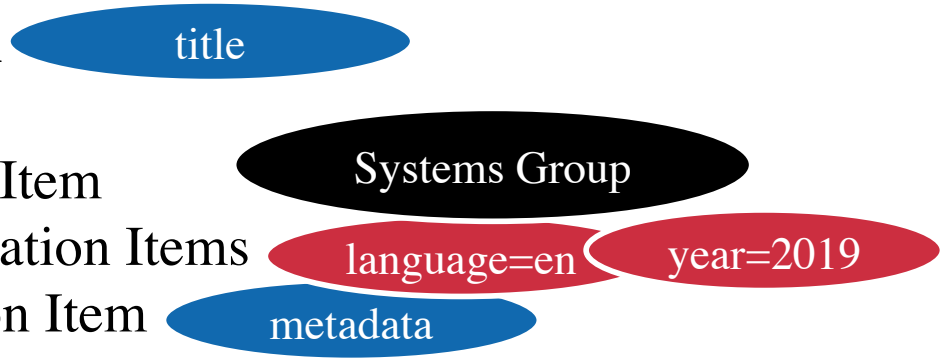
Element Information Items

Element Information Item metadata
[local name] metadata
[children] Element Information Items title publisher
[attributes] <empty>
[parent] Document Information Item doc

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE metadata>
<metadata>
    <title
        language="en"
        year="2019"
        >Systems Group</title>
    <publisher>ETH Zurich</publisher>
</metadata>
```

Element Information Items

Element Information Item
[local name] title
[children] Text Information Item
[attributes] Attribute Information Items
[parent] Element Information Item




```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE metadata>
<metadata>
    <title
        language="en"
        year="2019"
        >Systems Group</title>
    <publisher>ETH Zurich</publisher>
</metadata>
```

Element Information Items

Element Information Item publisher
[local name] publisher
[children] Text Information Items ETH Zurich
[attributes] Attribute Information Items <empty>
[parent] Element Information Item dc:metadata

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE metadata>
<metadata>
    <title
        language="en"
        year="2019"
        >Systems Group</title>
    <publisher>ETH Zurich</publisher>
</metadata>
```

Attribute Information Items

Attribute Information Item year=2019
[local name] year
[normalized value] 2019
[owner element] Element Information Item title

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE metadata>
<metadata>
    <title
        language="en"
        year="2019"
        >Systems Group</title>
    <publisher>ETH Zurich</publisher>
</metadata>
```

Attribute Information Items

Attribute Information Item language=en
[local name] language
[normalized value] en
[owner element] Element Information Item title

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE metadata>
<metadata>
    <title
        language="en"
        year="2019"
        >Systems Group</title>
    <publisher>ETH Zurich</publisher>
</metadata>
```

Text Information Items

Text Information Item Systems Group

[characters] S y s t e m s <space> G r o u p

[owner element] Element Information Item title

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE metadata>
<metadata>
    <title
        language="en"
        year="2019"
        >Systems Group</title>
    <publisher>ETH Zurich</publisher>
</metadata>
```

Text Information Items

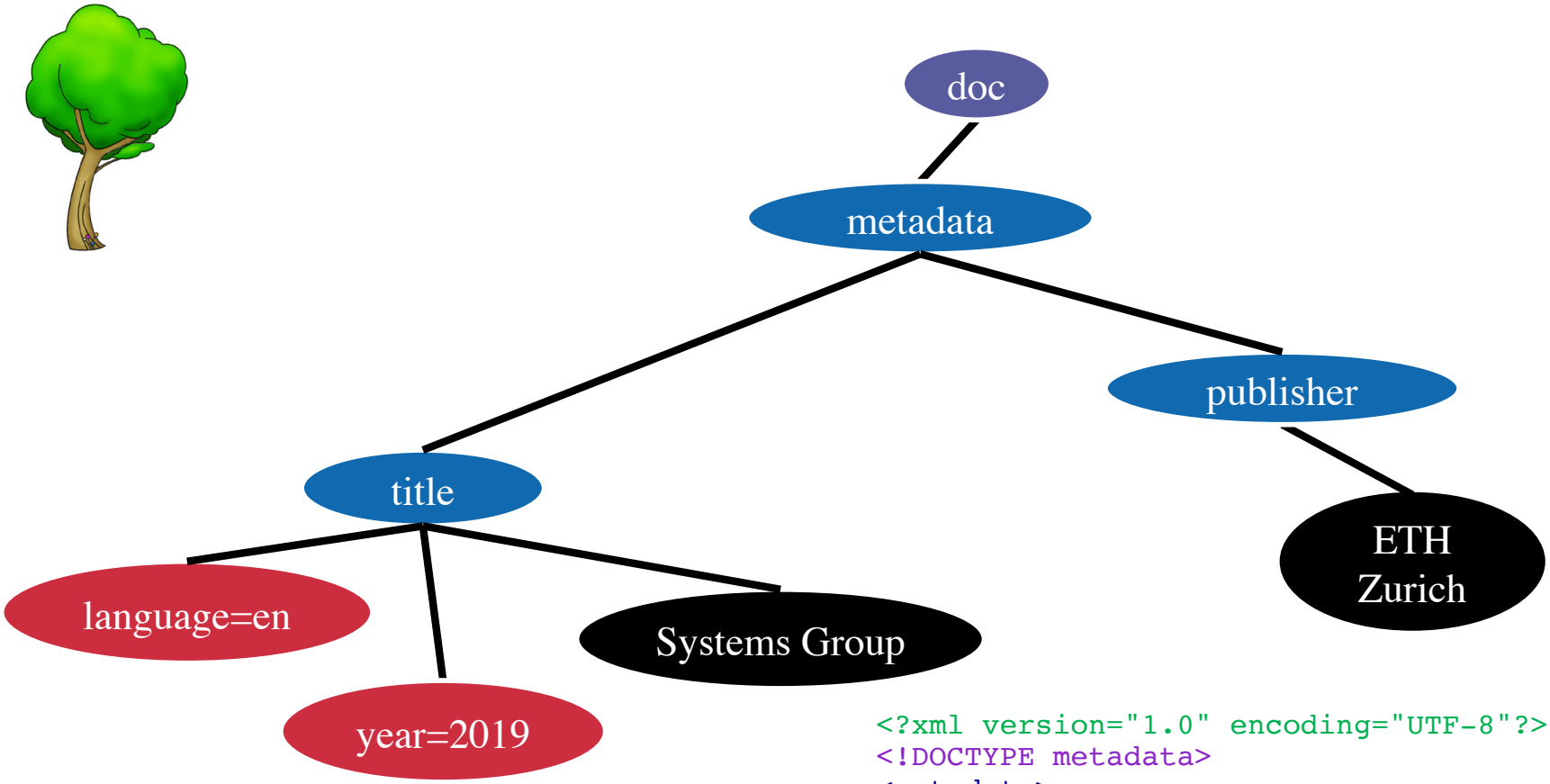
Text Information Item ETH Zurich

[characters] E T H <space> Z u r i c h

[owner element] Element Information Item publisher

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE metadata>
<metadata>
    <title
        language="en"
        year="2019"
        >Systems Group</title>
    <publisher>ETH Zurich</publisher>
</metadata>
```

XML Infoset - the tree




```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE metadata>
<metadata>
    <title
        language="en"
        year="2019"
        >Systems Group</title>
    <publisher>ETH Zurich</publisher>
</metadata>
```

Types

Type Systems

Almost **all type systems** (Java, SQL, PSVI, JDM, Protocol buffers, Avro, Parquet, and so on) share the following properties:

- Distinction between **atomic** types and **structured** types

- **Same categories** of atomic types

- **Lists** and **maps** as structured types

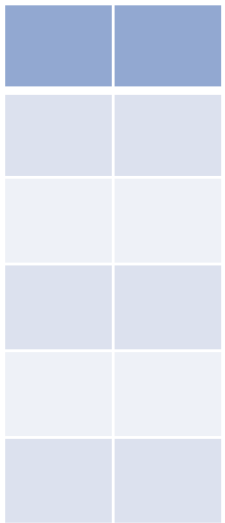
- Sequence type **cardinalities**

Types (General)

- Atomic Types

vs.

Structured Types



Atomic Types

Atomic Types

Strings

Strings
(Character sequences with monoid structure)

"foo" → f o o

"Zurich" → Z u r i c h

"Ilsebill salzte nach." → I l s e b i l l ␣ s a l z t e ␣ n a c h .

Atomic Types

Strings
Numbers

Interval-based integer types (exist as signed and unsigned)

8-bit (Java's byte)
16-bit (Java's short)
32-bit (Java's int)
64-bit (Java's long)

Arbitrary precision decimals (and integers)

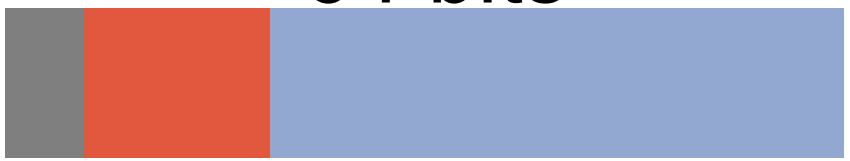
Any precision and scale

3141592653.589793238462643383279 5

Float and Double IEEE 754 standard

32 bits

64 bits



ca. 7 digits

3141592000

ca. 15 digits

3141592653.58979

10^{-37} to 10^{37}

10^{-307} to 10^{308}

single precision

double precision

Atomic Types

Strings
Numbers
Booleans

Booleans

TRUE
t
true
y
yes
on
1

FALSE
f
false
n
no
off
0

Atomic Types

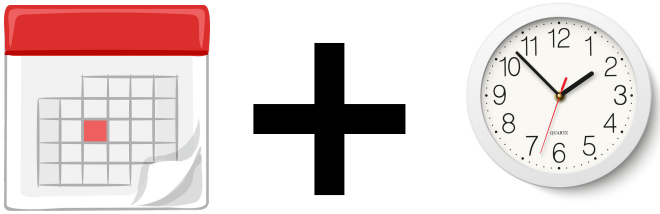
Strings
Numbers
Booleans
Dates and Times

Dates and times

Date

Time

Timestamp



Duration

Dates (Gregorian calendar)

Year + Month + Day

2017 August 1st

(AD)

Times

Hours + Minutes + Seconds

10 : 31 : 15.109378

Timestamps

Year + Month + Day + Hours + Minutes + Seconds

2017 August 1st 10 : 31 : 15.109378
(AD)

Atomic Types

Strings
Numbers
Booleans
Dates and Times
Time Intervals

Duration kinds

Year	Month	Day	Hour	Minute	Second
		Example: 2 years and 4 months			

Duration kinds

Atomic Types

Strings
Numbers
Booleans
Dates and Times
Time Intervals
Binaries

Atomic Types

Strings
Numbers
Booleans
Dates and Times
Time Intervals
Binaries
Null

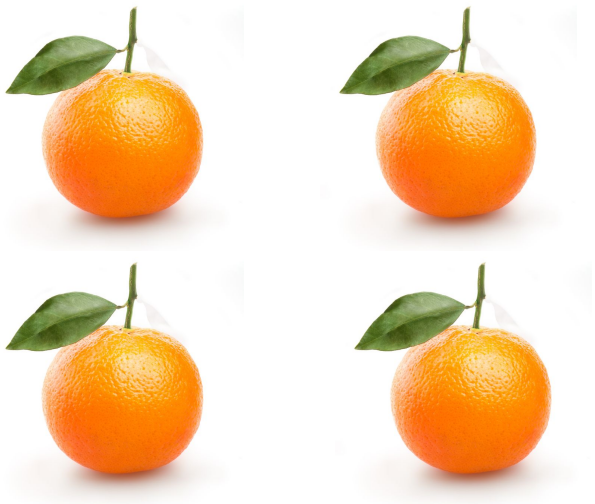
Lexical space vs. value space

"1"
"01"
...

Value space

Lexical space

Lexical space vs. value space

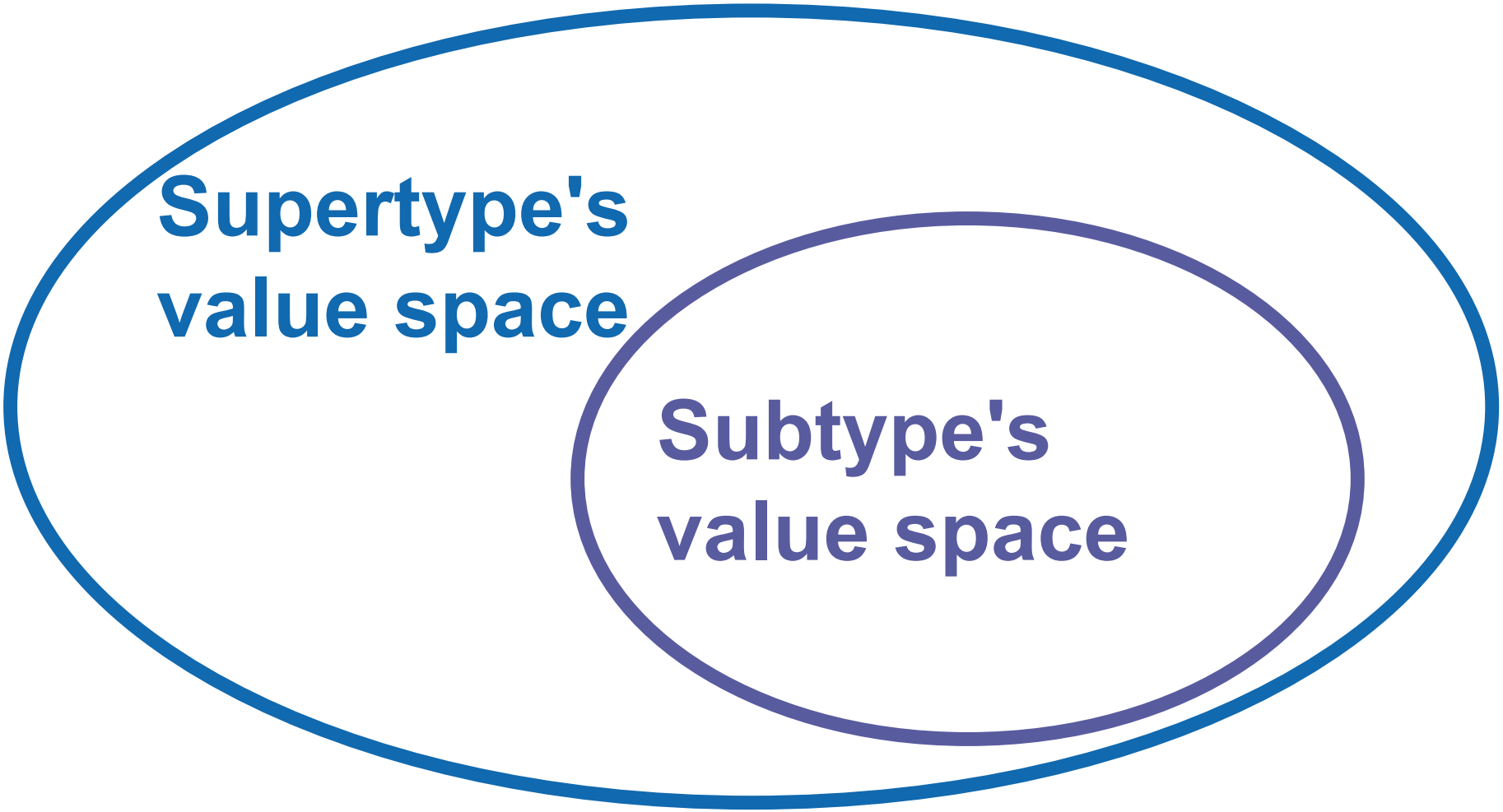


"4"
"+4E00"
"100b"
...

Value space

Lexical space

Subtypes



Structured Types

Data Structure
Maps (Key-value model!)
Lists

Structured Types

Data Structure	Examples
Maps (Key-value model!)	JSON Object, Set of XML Attributes Protobuf Message,
Lists	JSON Array, XML Element, Protobuf repeated field

Cardinality

How many?	Common sign	Common adjective

Cardinality

How many?	Common sign	Common adjective
One		required

Cardinality

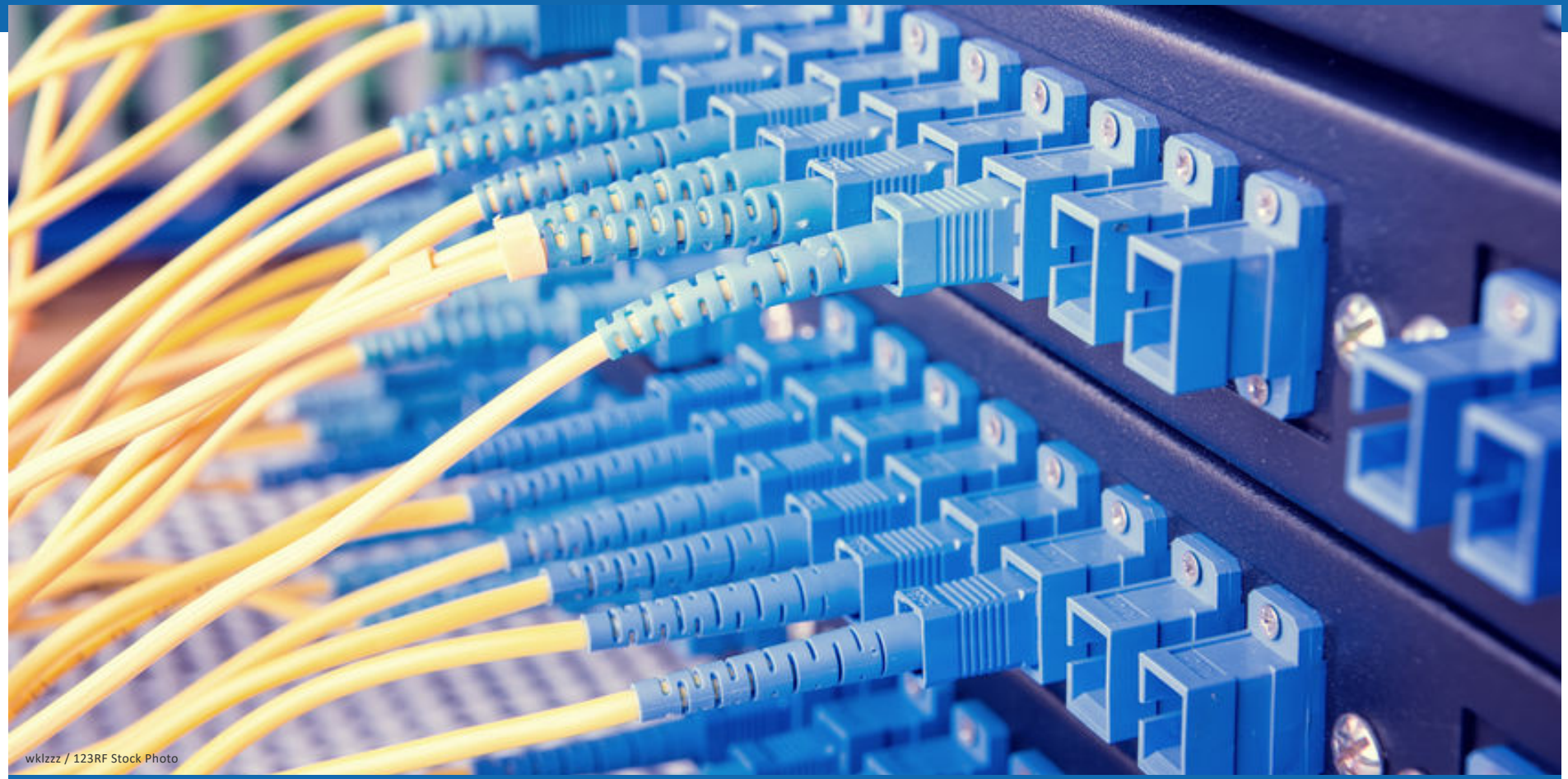
How many?	Common sign	Common adjective
One		required
Zero or more	*	repeated

Cardinality

How many?	Common sign	Common adjective
One		required
Zero or more	*	repeated
Zero or one	?	optional

Cardinality

How many?	Common sign	Common adjective
One		required
Zero or more	*	repeated
Zero or one	?	optional
One or more	+	



Protocol Buffers

Messages

```
message Person {
  required string last_name = 1;
  repeated string first_name = 2;
  optional Title title = 3;
  optional Person boss = 4;
}
```

Scalar types

double, float
int32, int64 and variants
bool
string
bytes

Enums

```
enum Title {
  MR = 1;
  MS = 2;
  MRS = 3;
}
```

In C++

```
person.boss().first_name()
```

This is a schema!

```
message Person {
  required string last_name = 1;
  repeated string first_name = 2;
  optional Title title = 3;
  optional Person boss = 4;
}
```

JSON/XML vs. Protobufs

No schema
(heterogeneous)

With schema
(homogeneous)

Without schema...

```
{
  "a" : 1,
  "b" : [ "foo", true, null, { "foo" : "bar" } ],
  "c" : {
    "d" : { "foo" : null },
    "e" : [ 1, 2, [ 3, 4 ] ],
    "f" : 3.14
  }
}
```

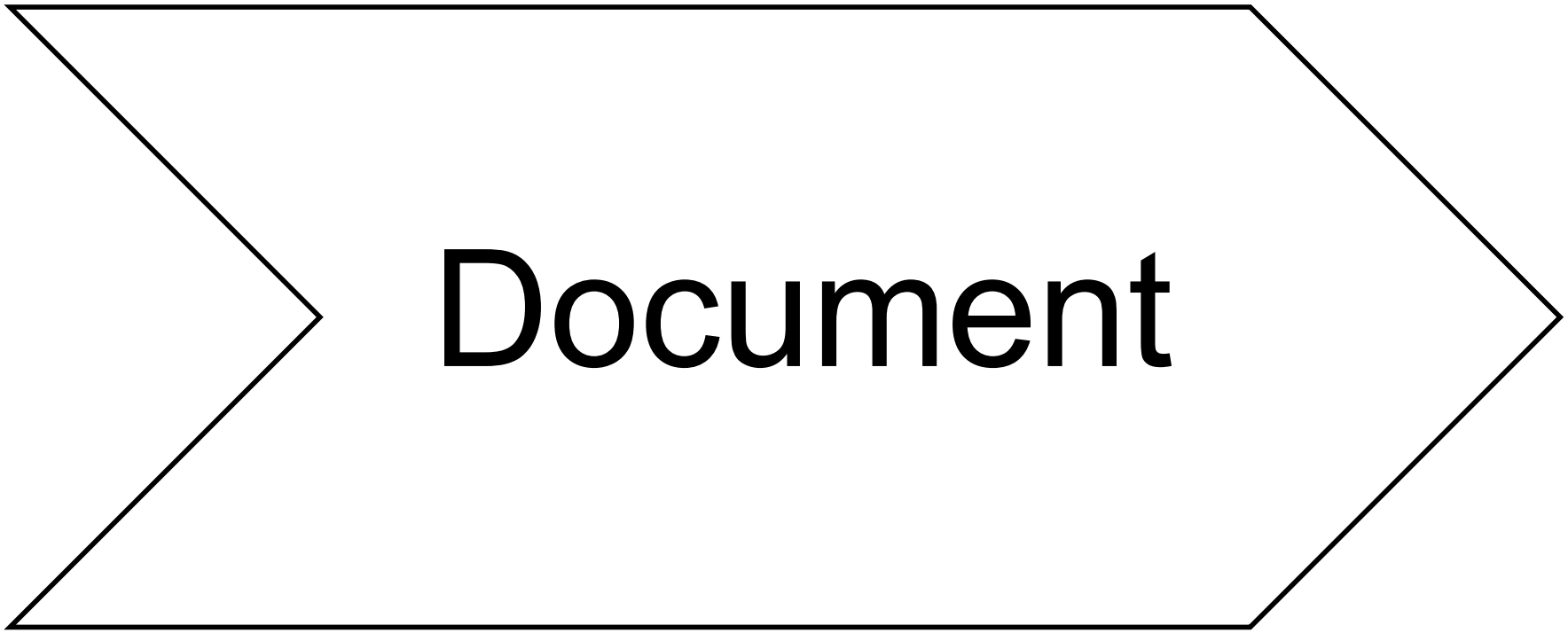
With schema...

```
{
  "a" : 1,
  "b" : true,
  "c" : [
    { "foo" : "bar1",  "bar" : [ 1, 2 ] },
    { "foo" : "bar2",  "bar" : [ 3, 4, 5 ] },
    { "foo" : "bar3" }
  ]
}
```

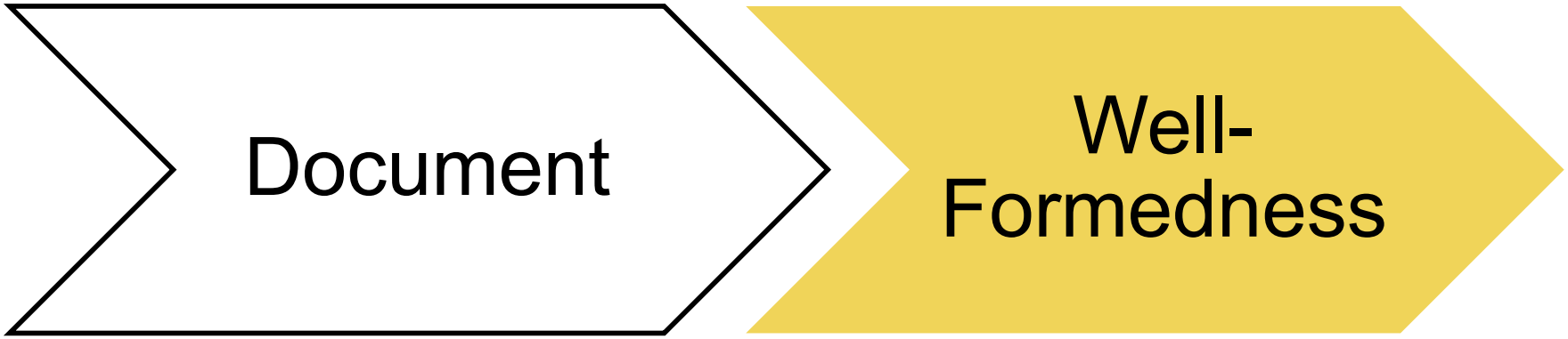
Burak Cakmak / 123RF Stock Photo

Validation

Validation: The Pipeline

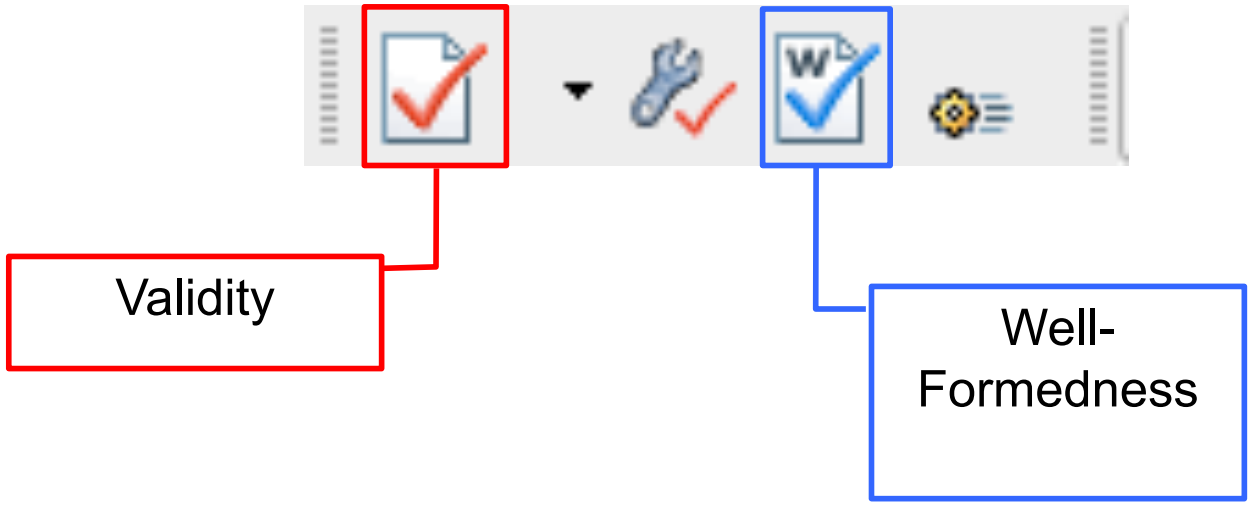


Validation: The Pipeline



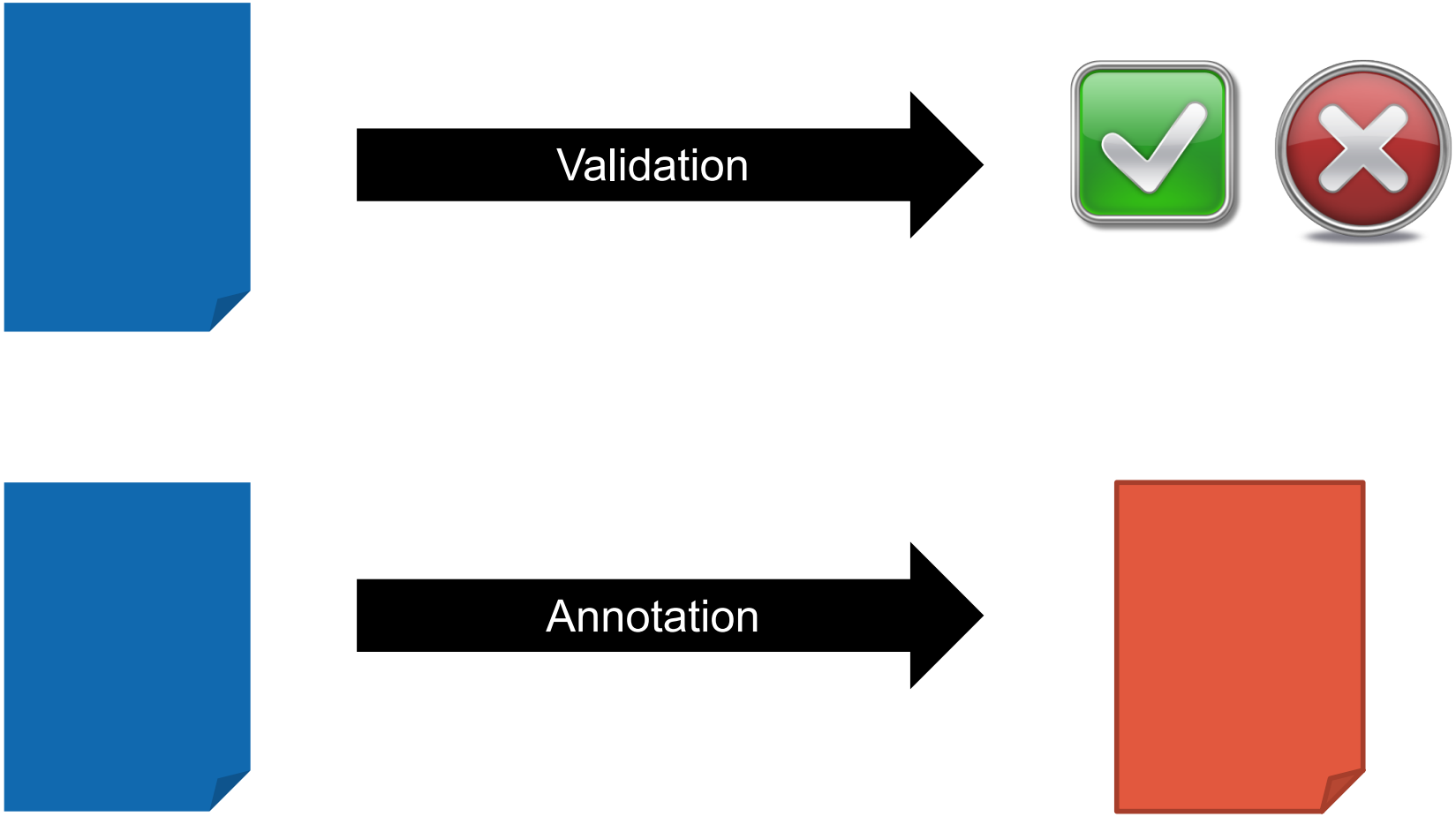
Validation: The Pipeline

On the oXygen Cheat Sheet



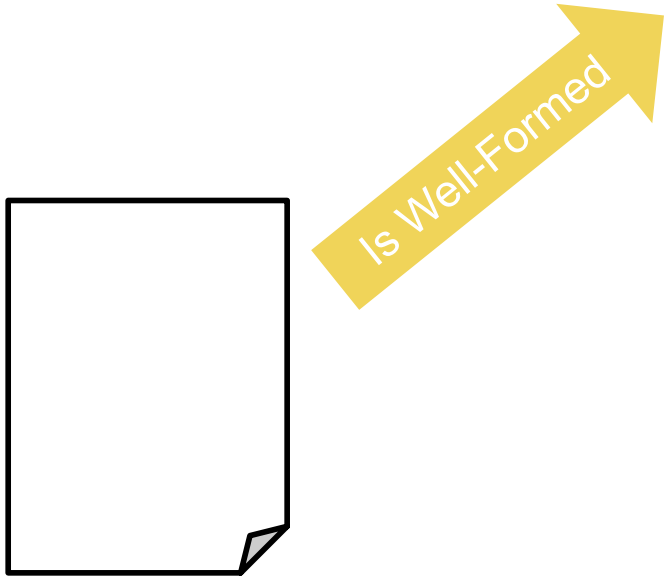
Validation vs. Annotation

Validation vs. Annotation

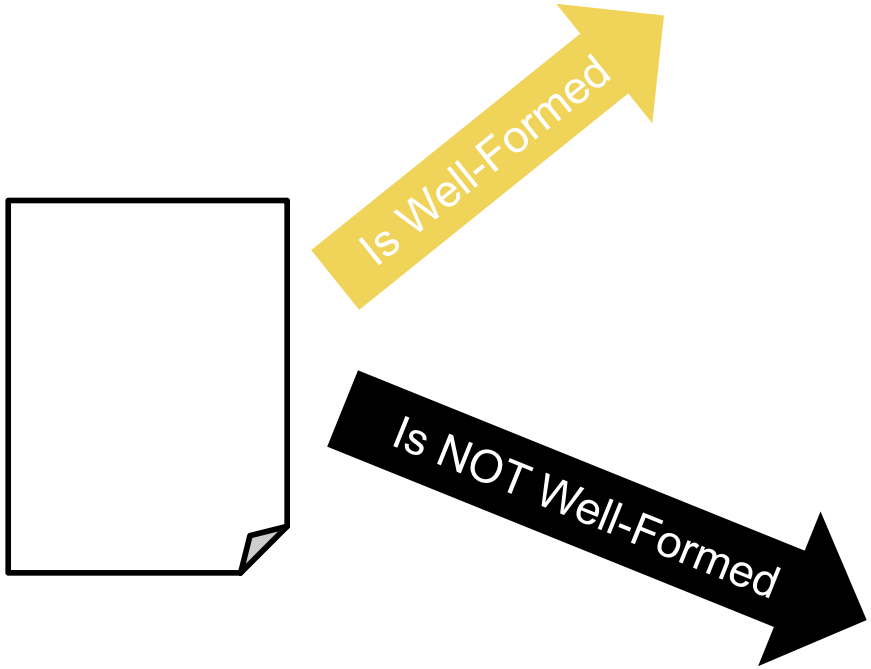


Validation

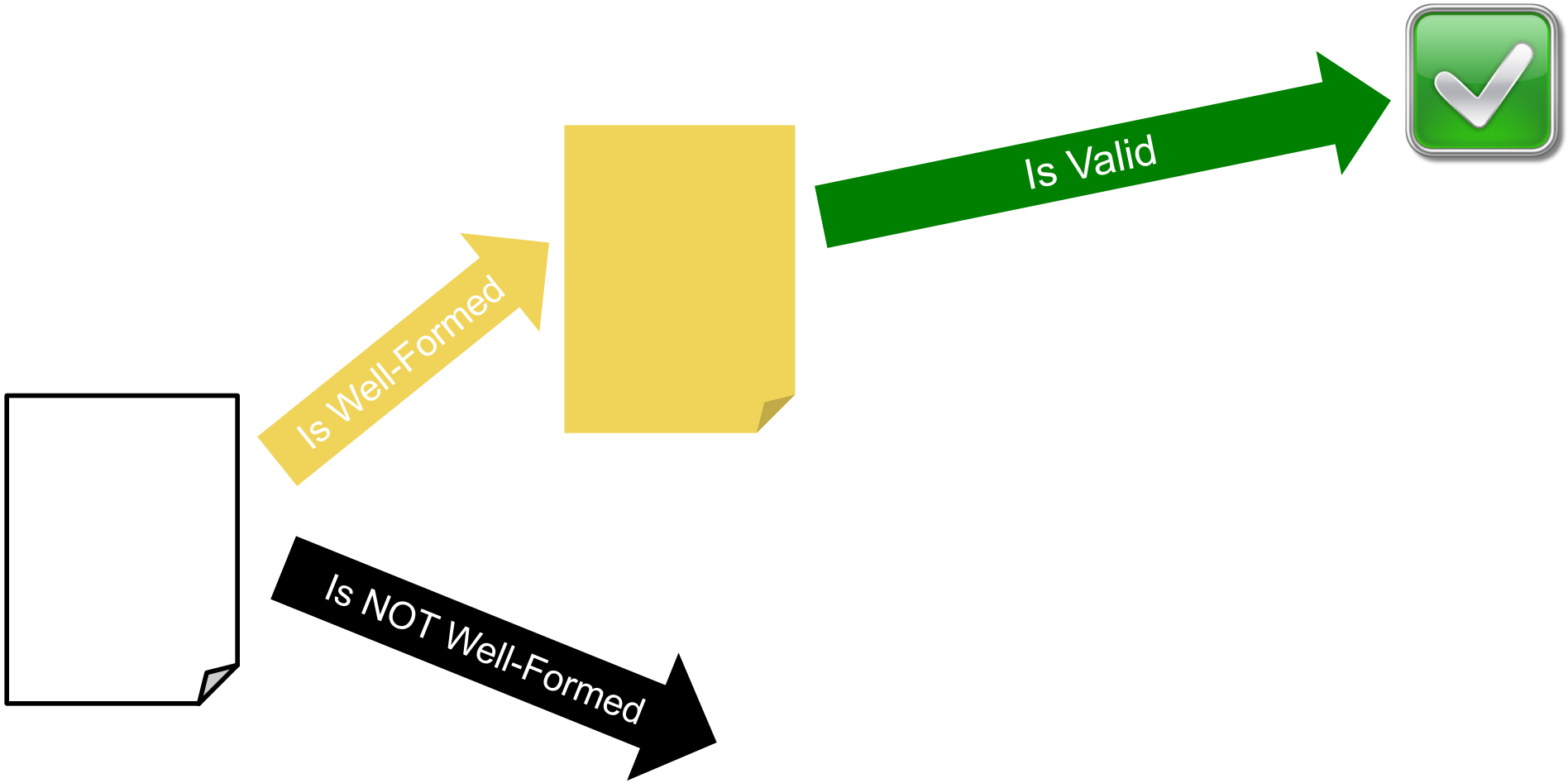
Validation



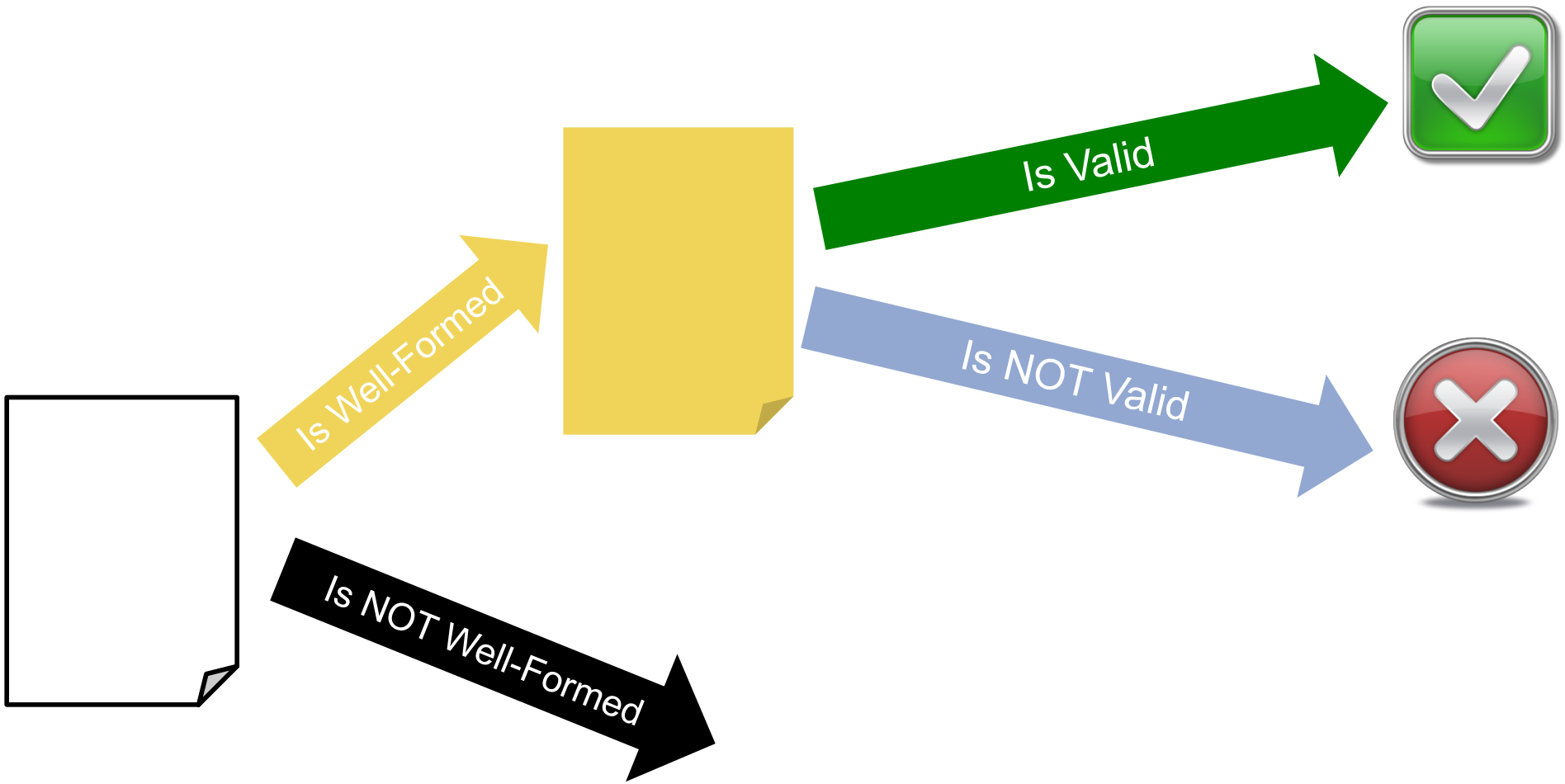
Validation



Validation



Validation



DTD Validation (just to know what it looks like)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE a [
<!ELEMENT a (foo+, bar*, foobar?)>
<!ELEMENT foo EMPTY>
<!ELEMENT bar EMPTY>
<!ELEMENT foobar EMPTY>
]>
<a>
  <foo/>
  <foo/>
  <foo/>
  <foobar/>
</a>
```

XML Schema

Example XML document

```
<?xml version="1.0" encoding="UTF-8"?>
<foo>
  This is text.
</foo>
```

Example Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="foo" type="xs:string"/>
</xs:schema>
```

Simple Scenario

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="foo" type="xs:string"/>
</xs:schema>
```

schema.xsd

Schema

Instance

file.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<foo
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="schema.xsd">
  This is text.
</foo>
```

Simple Scenario

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="foo" type="xs:string"/>
</xs:schema>
```

schema.xsd

Schema

Instance

file.xml

```
<?xml version="1.0" enco    ="UTF-8"?>
<foo
  xmlns:xsi="http://www.    rg/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="schema.xsd">
  This is text.
</foo>
```

Simple Scenario

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="foo" type="xs:string"/>
</xs:schema>
```

Schema

Instance

```
<?xml version="1.0" encoding="UTF-8"?>
<foo
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="schema.xsd">
  This is text.
</foo>
```

Simple Scenario

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="foo" type="xs:integer"/>
</xs:schema>
```

Schema

Instance

```
<?xml version="1.0" encoding="UTF-8"?>
<foo
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="schema.xsd">
  142857
</foo>
```

Simple Types: Built-in

Strings	**string** **anyURI** **QName**
Numbers	**decimal** integer **float** **double** long int short byte positiveInteger nonNegativeInteger... unsignedLong unsignedInt...
Booleans	**boolean**

Simple Types: Built-in

Dates and Times	**dateTime** **time** **date** **gYearMonth** **gMonthDay** **gYear** **gMonth** **gDay** dateTimeStamp
Time Intervals	**duration** yearMonthDuration dayTimeDuration
Binaries	**hexBinary base64Binary**
Null	-

Dates

2014-12-02

2014-12-02T10:15:00Z

01:15:00-08:00

Durations

P1Y2MT3H

User-defined types

User-defined types

Restriction

User-defined types

Restriction

Union Not atomic

User-defined types

Restriction

Union Not atomic

List Not atomic

Restriction

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:simpleType name="myFixedLengthString">
    <xs:restriction base="xs:string">
      <xs:length value="3"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="foo" type="myFixedLengthString"/>
</xs:schema>
```

Schema

Instance

```
<?xml version="1.0" encoding="UTF-8"?>
<foo
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="schema.xsd">ZRH</foo>
```

Restriction

```
<xs:simpleType name="myFixedLengthString">
  <xs:restriction base="xs:string">
    <xs:length value="3"/>
  </xs:restriction>
</xs:simpleType>
```

Schema

Instance

```
<foo>ZRH</foo>
```

List

```
<xs:simpleType name="myList">
  <xs:list itemType="xs:string"/>
</xs:simpleType>
```

Schema

Instance

```
<foo>foo bar foobar</foo>
```

Union

```
<xs:simpleType name="myUnion">
  <xs:union memberTypes="xs:integer xs:boolean"/>
</xs:simpleType>
```

Schema

Instance

```
<foo>true</foo>
```

Complex Types

Complex Types

Empty

<foo/>

Complex Types

Empty <foo/>

Simple Content <foo>text</foo>

Complex Types

Empty

```
<foo/>
```

Simple Content

```
<foo>text</foo>
```

Complex Content

```
<foo>
  <a/>
  <b/>
</foo>
```

Complex Types

Empty

```
<foo/>
```

Simple Content

```
<foo>text</foo>
```

Complex Content

```
<foo>
  <a/>
  <b/>
</foo>
```

Mixed Content

```
<foo>
  Text<a/>Text<b/>
</foo>
```

Complex content

```
<xs:complexType name="complexContent">
  <xs:sequence>
    <xs:element name="twotofour" type="xs:string" minOccurs="2" maxOccurs="4"/>
    <xs:element name="zeroorone" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>
```

Schema

Instance

```
<foo>
  <twotofour>foobar</twotofour>
  <twotofour>foobar</twotofour>
  <twotofour>foobar</twotofour>
  <zeroorone>true</zeroorone>
</foo>
```

Complex content

```
<xs:complexType name="complexContent">
  <xs:sequence>
    <xs:element name="twotofour" type="xs:string" minOccurs="2" maxOccurs="4"/>
    <xs:element name="zeroorone" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>
```

Schema

Instance

```
<foo>
  <twotofour>foobar</twotofour>
  <twotofour>foobar</twotofour>
  <twotofour>foobar</twotofour>
  <zeroorone>true</zeroorone>
</foo>
```

Empty content

```
<xs:complexType name="emptyType">
  <xs:sequence/>
</xs:complexType>
```

Schema

Instance

```
<foo/>
```

Simple content

```
<xs:complexType name="dateCountry">
  <xs:simpleContent>
    <xs:extension base="xs:date">
      <xs:attribute name="country" type="xs:string"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

Schema

Instance

```
<foo country="Switzerland">2014-12-02</foo>
```

Mixed content

```
<xs:complexType name="mixedContent" mixed="true">
  <xs:sequence>
    <xs:element name="b" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

Schema

Instance

```
<foo>Some text and some <b>bold</b> text.</foo>
```

Simple type on attributes

```
<xs:complexType name="withAttribute">
  <xs:sequence/>
  <xs:attribute name="country"
                type="xs:string"
                default="Switzerland"/>
</xs:complexType>
```

Schema

Instance

```
<foo country="Switzerland"/>
```

Named Types

```
<xs:complexType name="empty">
<xs:sequence/>
</xs:complexType>

<xs:element name="c" type="empty">
</xs:element>
```

Schema

Instance

```
<c/>
```

Anonymous Types

```
<xs:element name="c">
  <xs:complexType>
    <xs:sequence/>
  </xs:complexType>
</xs:element>
```

Schema

Instance

```
<c/>
```

No namespaces

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  <xs:element name="foo" type="xs:string"/>
</xs:schema>
```

Schema

Instance

```
<?xml version="1.0" encoding="UTF-8"?>
<foo
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="schema.xsd">
  This is text.
</foo>
```

Keys

```
<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="root">
    <xs:complexType>
      ..
    </xs:complexType>
    <xs:key name="foo-id">
      <xs:selector xpath="foo"/>
      <xs:field xpath="@id"/>
    </xs:key>
  </xs:element>
</xs:schema>

<?xml version="1.0" encoding="UTF-8"?>
<root
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="schema.xsd">
  <foo id="foo"/>
  <foo id="bar"/>
  <foo id="foobar"/>
</root>
```

Keys

```
<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="root">
    <xs:complexType>
      ..
    </xs:complexType>
    <xs:key name="foo-id">
      <xs:selector xpath="foo"/>
      <xs:field xpath="@id"/>
    </xs:key>
  </xs:element>
</xs:schema>

<?xml version="1.0" encoding="UTF-8"?>
<root
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="schema.xsd">
  <foo id="foo"/>
  <foo id="bar"/>
  <foo id="foobar"/>
</root>
```

What **must** be unique

Keys

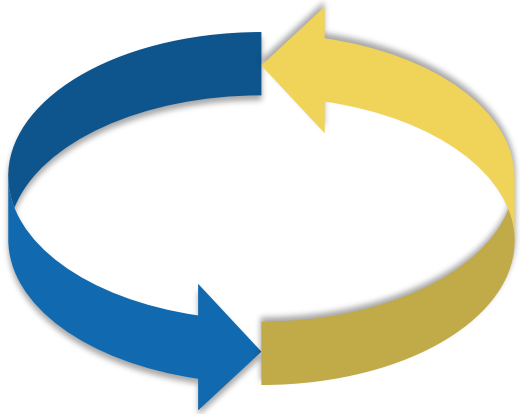
```
<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="root">
    <xs:complexType>
      ..
    </xs:complexType>
    <xs:key name="foo-id">
      <xs:selector xpath="foo"/>
      <xs:field xpath="@id"/>
    </xs:key>
  </xs:element>
</xs:schema>

<?xml version="1.0" encoding="UTF-8"?>
<root
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="schema.xsd">
  <foo id="foo"/>
  <foo id="bar"/>
  <foo id="foobar"/>
</root>
```

What **must** be unique

What **makes** it unique

Bonus material: The Schema of Schemas

```
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.w3.org/2001/XMLSchema">
  <xs:element name="schema" id="schema">
    <xs:complexType>
      <xs:complexContent>
        ..
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="element" type="xs:topLevelElement" id="element"/>
  <xs:element name="simpleType" type="xs:topLevelSimpleType" id="simpleType"/>
  <xs:element name="complexType" type="xs:topLevelComplexType" id="complexType"/>
  <xs:complexType name="element" abstract="true">
    <xs:complexContent>
      ..
    </xs:complexContent>
  </xs:complexType>
</xs:schema>
```



JSON Schema

```
{
  "$id": "https://example.com/geographical-location.schema.json",
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "Longitude and Latitude Values",
  "description": "A geographical coordinate.",
  "required": [ "latitude", "longitude" ],
  "type": "object",
  "properties": {
    "latitude": { "type": "number", "minimum": -90, "maximum": 90 },
    "longitude": { "type": "number", "minimum": -180, "maximum": 180 }
  }
}
```

```
{
  "latitude": 48.858093,
  "longitude": 2.294694
}
```

JSound

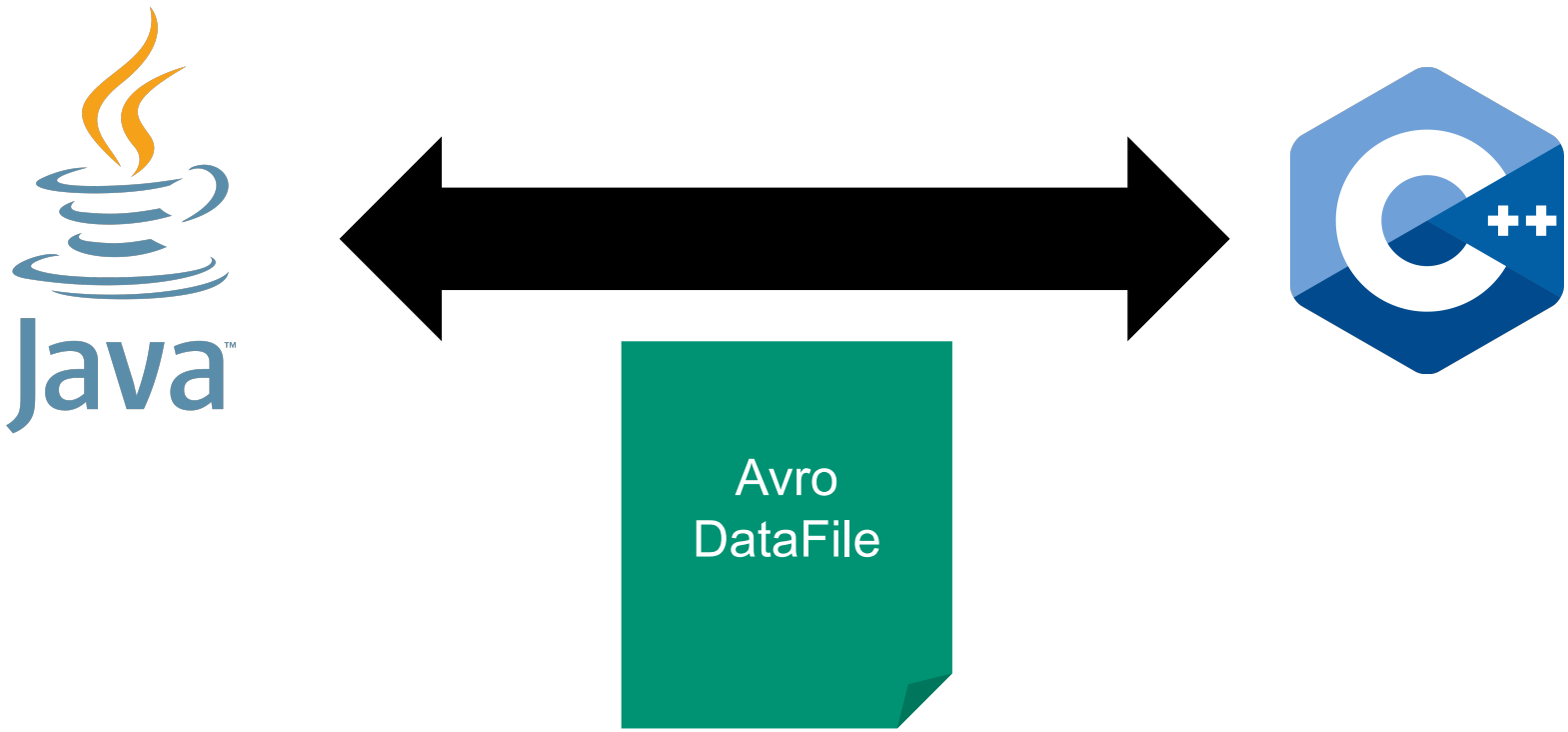
```
{
  "person" : {
    "@lastName" : "string",
    "!firstNames" : [ "string" ],
    "age?" : "integer"
  }
}
```

```
{
  "lastName": "Doe",
  "firstNames": [ "John", "James" ],
  "age": null
}
```

Avro

Avro is language neutral

Interoperability



Avro atomic types

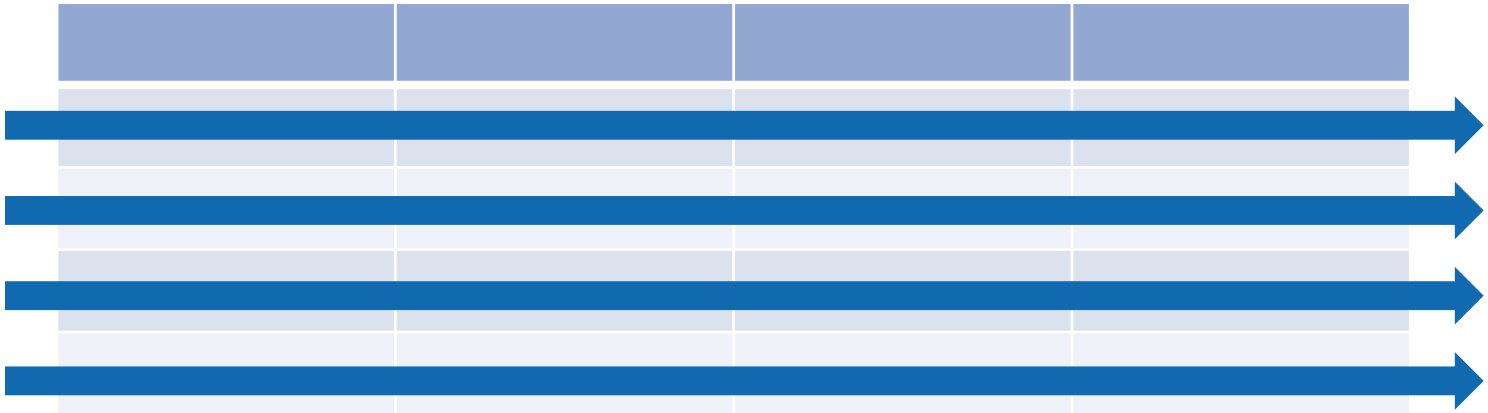
Category	Types
Absence of value	`null`
Boolean	`boolean`
Number	`int`
	`long`
	`float`
	`double`
String	`string`
	`enum`
Binary	`bytes`
	`fixed (list of 8-bit unsigned bytes)`

Avro structured types

Category	Types
Arrays	`array`
Objects	`map`
	`record`

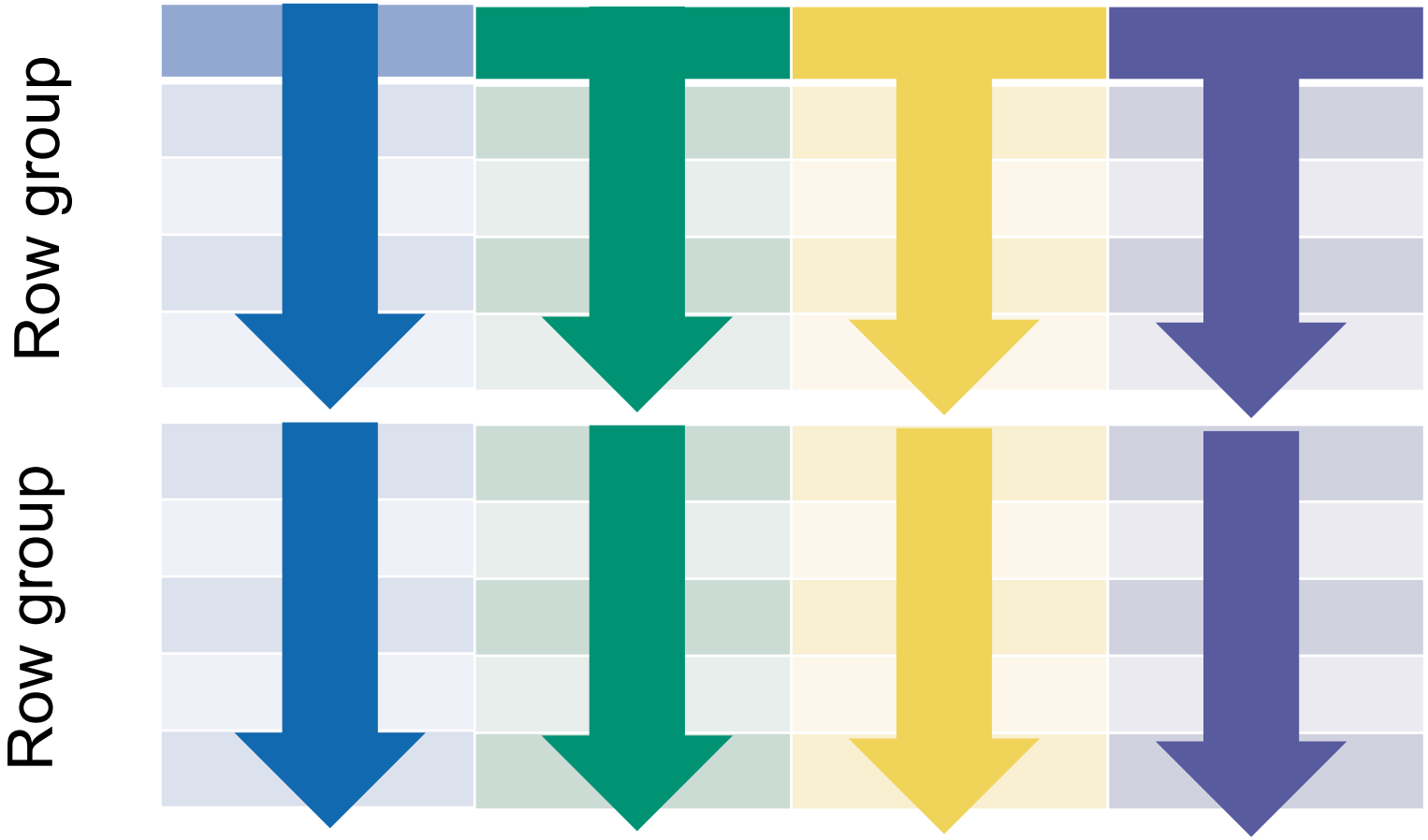
Parquet

Row storage



Columnar storage

Columnar storage



Columnar storage

Parquet atomic types

Category	Types
Boolean	boolean
Number	int32
	int64
	int96
	double
	DECIMAL(precision, scale)
String	UTF8
	ENUM
Binary	binary
	fixed_len_byte_array
Date	DATE

Parquet structured types

Category	Types
Arrays	`LIST`
Objects	`MAP`