

Introduction

# Introduction to XML

---

Lecture "XML in Communication Systems"  
Chapter 1

Dr.-Ing. Jesper Zedlitz  
Research Group for Communication Systems  
Dept. of Computer Science  
Christian-Albrechts-University in Kiel



# Acknowledgements

---

Informatik · CAU Kiel

- Gregor Engels  
University of Paderborn
- Matthew Langham  
s&n AG Paderborn

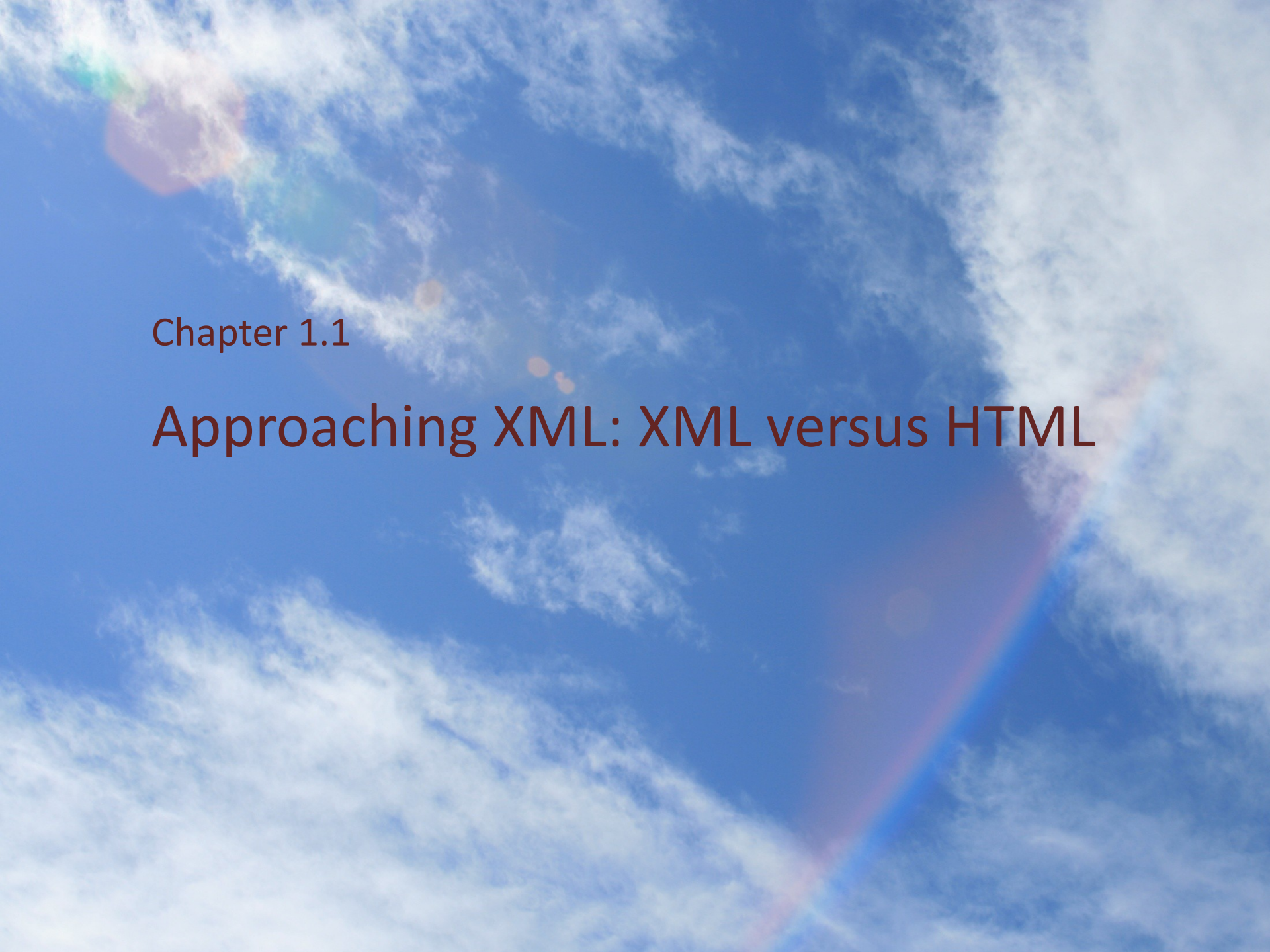
# Recommended Reading

- T. Bray, J. Paoli, C.M. Sperberg-McQueen, E. Maler, F. Yergeau, J. Cowan (Eds.):  
*XML 1.1 (Second Edition), W3C Recommendation, 16 August 2006.*  
<http://www.w3.org/TR/xml11>
- The Unicode Consortium:  
*The Unicode 5.0 Standard.*  
can be downloaded chapter-wise from  
<http://www.unicode.org/versions/Unicode5.0.0/>

# Overview

---

1. Approaching XML
  1. XML versus HTML
  2. Machine-to-machine communication
2. Well-formed XML documents
3. Excursion: Character sets
4. An initial XML shopping list

A vibrant blue sky with wispy white clouds and a prominent rainbow arching across the frame. The rainbow is a bright, multi-colored arc that spans from the lower left towards the upper right, with colors ranging from red to violet. The sky is a deep, clear blue, and the clouds are soft and white, scattered throughout the scene.

## Chapter 1.1

# Approaching XML: XML versus HTML

# XML versus HTML

- An HTML example

```
<h2>Inside XML</h2>
<p>
  <i>by
  <b>Holzner, St.</b> and
  <b>Else, Someone</b>
</i>
<br>
Indianapolis (New Riders) 2001
<br>
ISBN 0-7357-1020-1
```

- and the same example in XML

```
<book>
  <title>Inside XML</title>
  <author>Holzner, St.</author>
  <author>Else, Someone</author>
  <publisher>Indianapolis (New
Riders)
  </publisher>
  <year>2001</year>
  <ISBN>0-7357-1020-1</ISBN>
</book>
```

# HTML versus XML

---

- Similarities
  - Both use **tags**, e.g. <h2> and </year>
  - Tags may be **nested**: tags within tags
  - **Human** users can read and interpret both HTML and XML representations quite easily.

# XML versus HTML

- Another example

```
<h2>Relationship  
matter-energy</h2>  
<p>  
<i>E = M × c2</i>
```

- fixed set of tags
- presentation-oriented
- to be rendered  
via browser

- and the same example in XML

```
<equation>      <meaning>Relationship  
                  matter-energy</meaning>  
    <leftside>E</leftside>  
    <rightside>M × c2</rightside>  
</equation>
```

- extensible set of tags
- content-oriented
- transformation before  
rendering



# HTML versus XML

---

- HTML provides **typography-oriented** structuring
  - Document is cut into pieces alongside different "rules" for **typographical** representation
  - The main use of an HTML document is to **display** information.

"HTML is XML for typography."

- though "pure HTML" does not actually specify typographical features for different kinds of document pieces

# XML in Communications

- Automated interpretation of HTML?

```
<h2>Inside XML</h2>
<p>
  <i>by
    <b>Holzner, St.</b> and
    <b>Else, Someone</b>
  </i>
  <br>
  Indianapolis (New Riders) 2001
  <br>
  ISBN 0-7357-1020-1
```

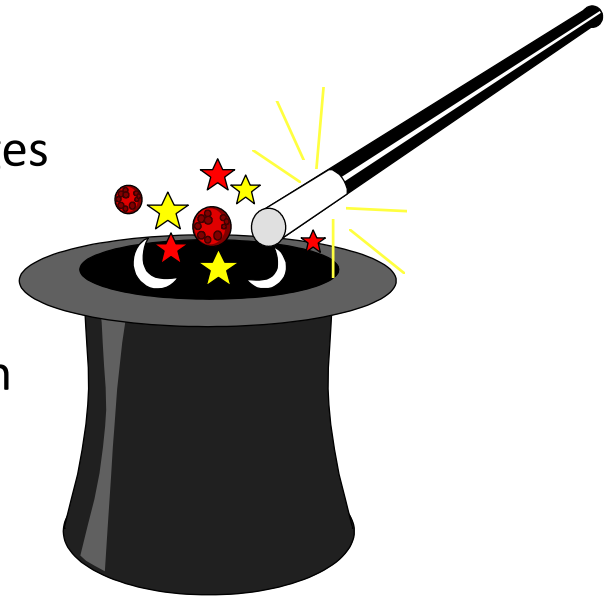
- Assume an (un)intelligent agent trying to retrieve the names of the authors of the book:
  - Authors' names appear immediately after the title
  - or immediately after the word "by"?
  - Are there two authors?
  - Or just one, called "Holzner, St. and Else, Someone"?

# HTML versus XML

- XML provides **content-oriented** structuring of information
  - The content of every "piece of information" is described: tags
  - Relations are defined through the nesting structure:  
e.g. the **author** element refers to the enclosing **book** element.
  - XML allows the definition of constraints on values
    - e.g. a year must be a number of four digits

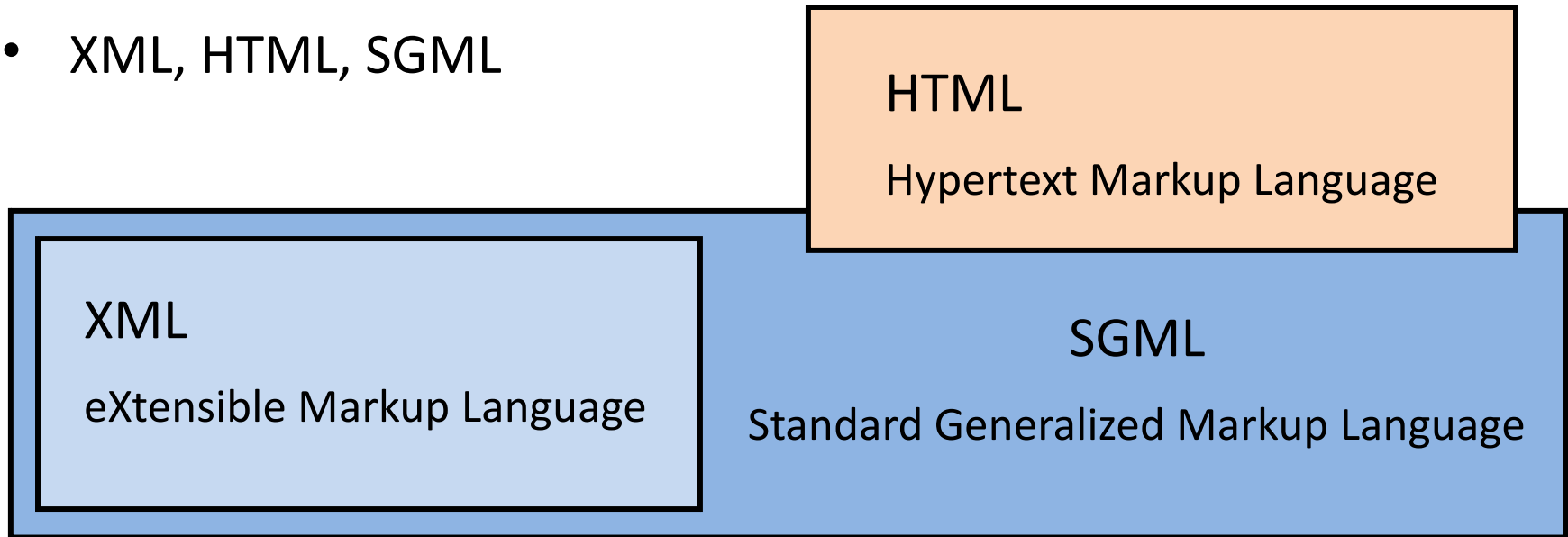
# XML versus HTML

- XML language technology
  - XML = eXtensible Markup Language:  
enables definition of domain-specific languages  
(vocabulary and grammar)
  - keeps distinction between  
internal structure and external representation
  - based on SGML, ISO standard since 1986  
(Standard Generalized Markup Language)
  - standardised by W3C
    - XML 1.0 in Feb. 1998
    - XML 1.1 in Aug. 2006



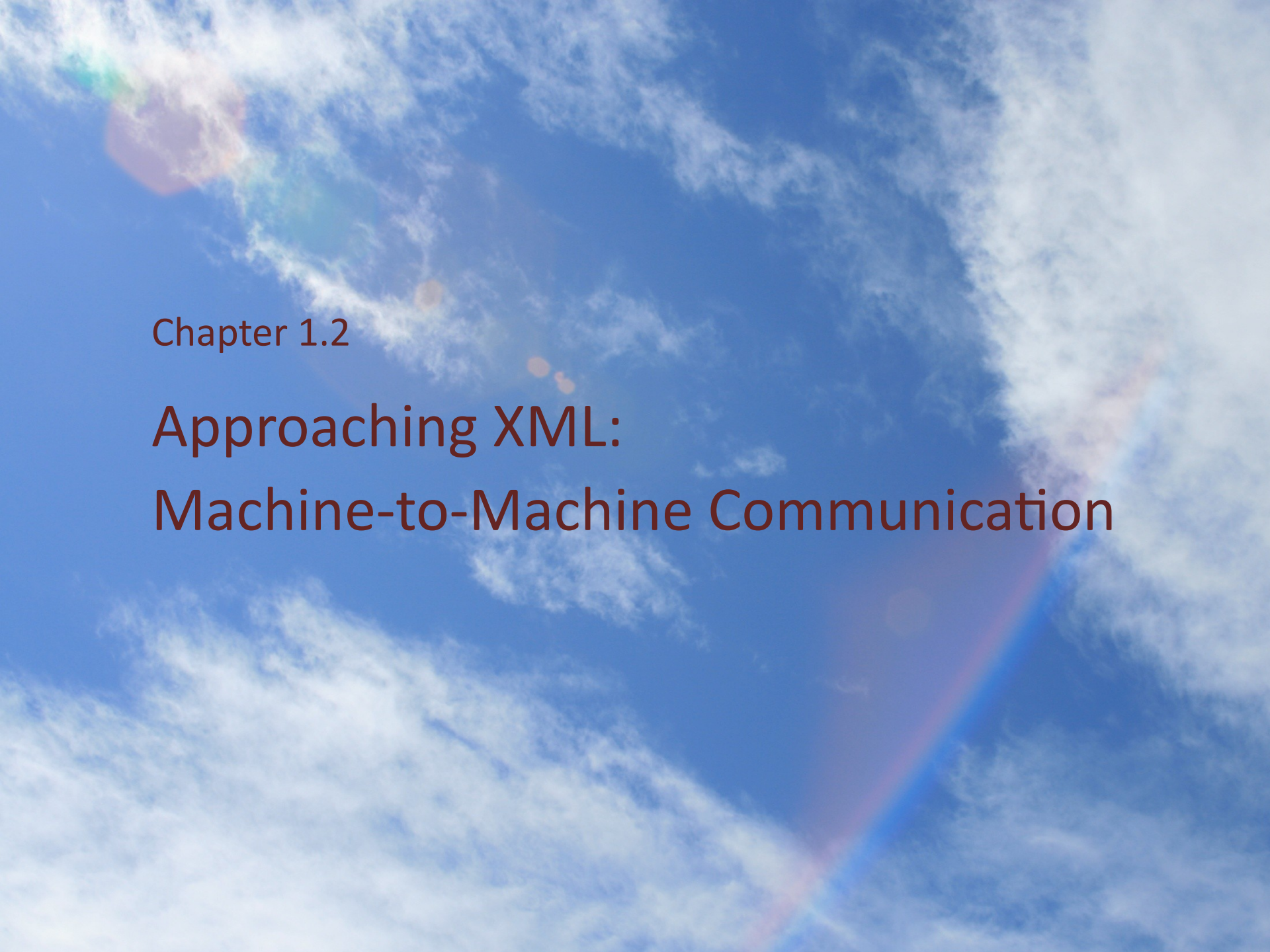
# XML versus HTML

- XML, HTML, SGML



- XML: a **subset** of SGML
- HTML: an **application** of SGML for the typography domain

$\text{HTML4.0} \in \text{XML} \subset \text{SGML}$

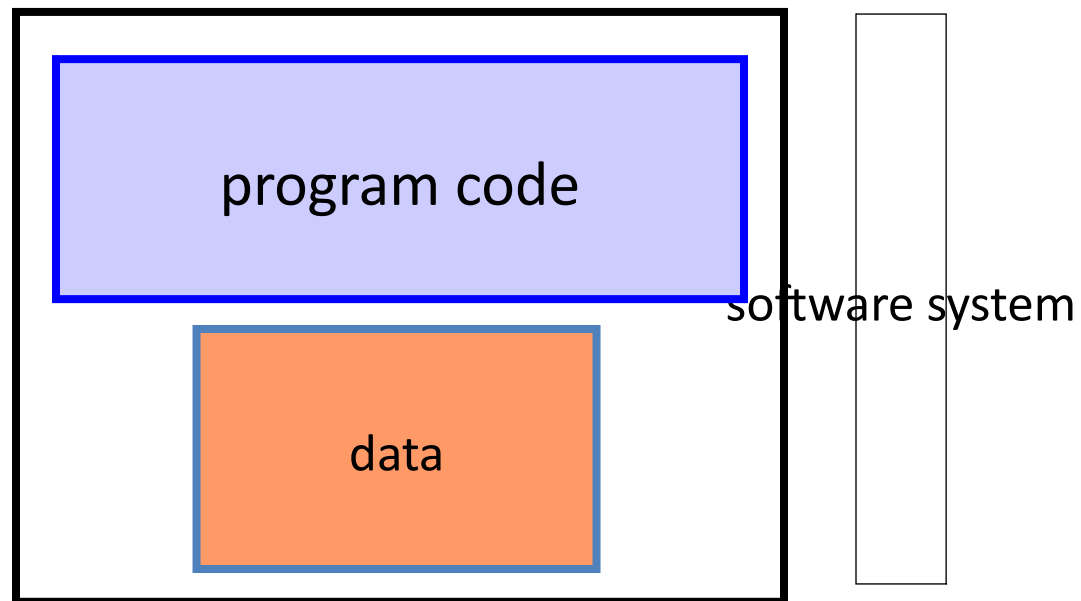
A vibrant blue sky filled with wispy white clouds. A bright rainbow arches across the lower right portion of the frame, its colors vivid against the blue background. In the upper left, there are some colorful lens flare artifacts.

Chapter 1.2

# Approaching XML: Machine-to-Machine Communication

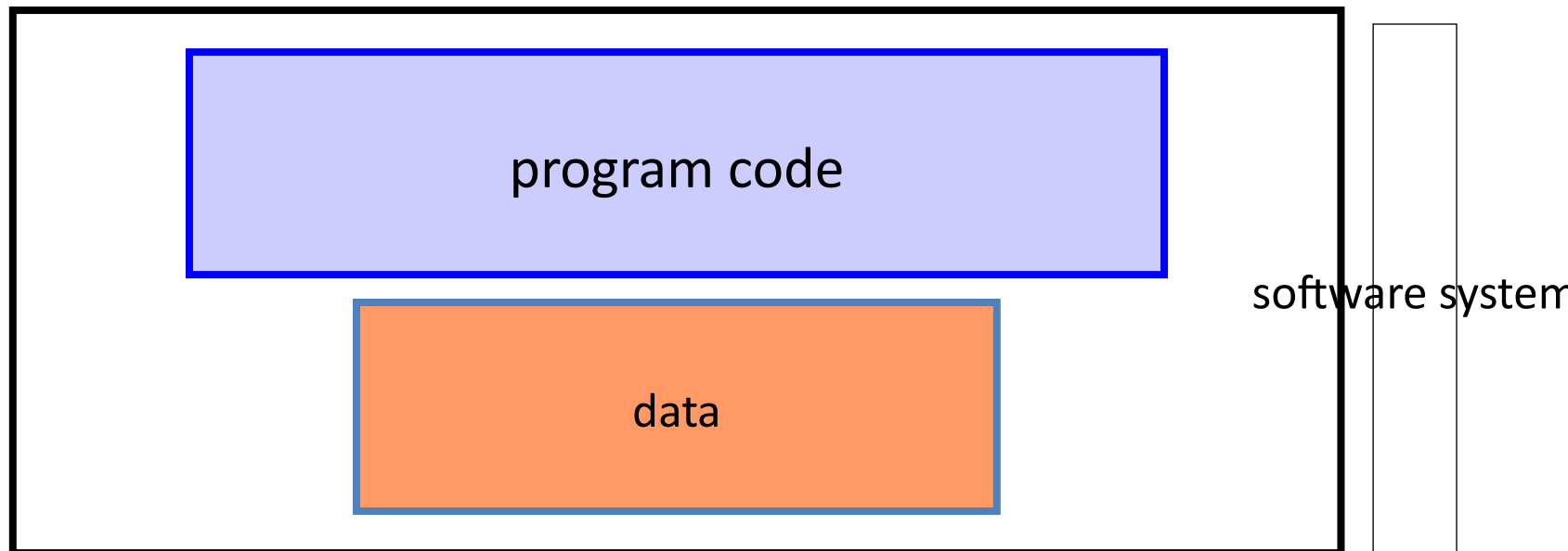
# XML in Communications

- In the beginning ...



# XML in Communications

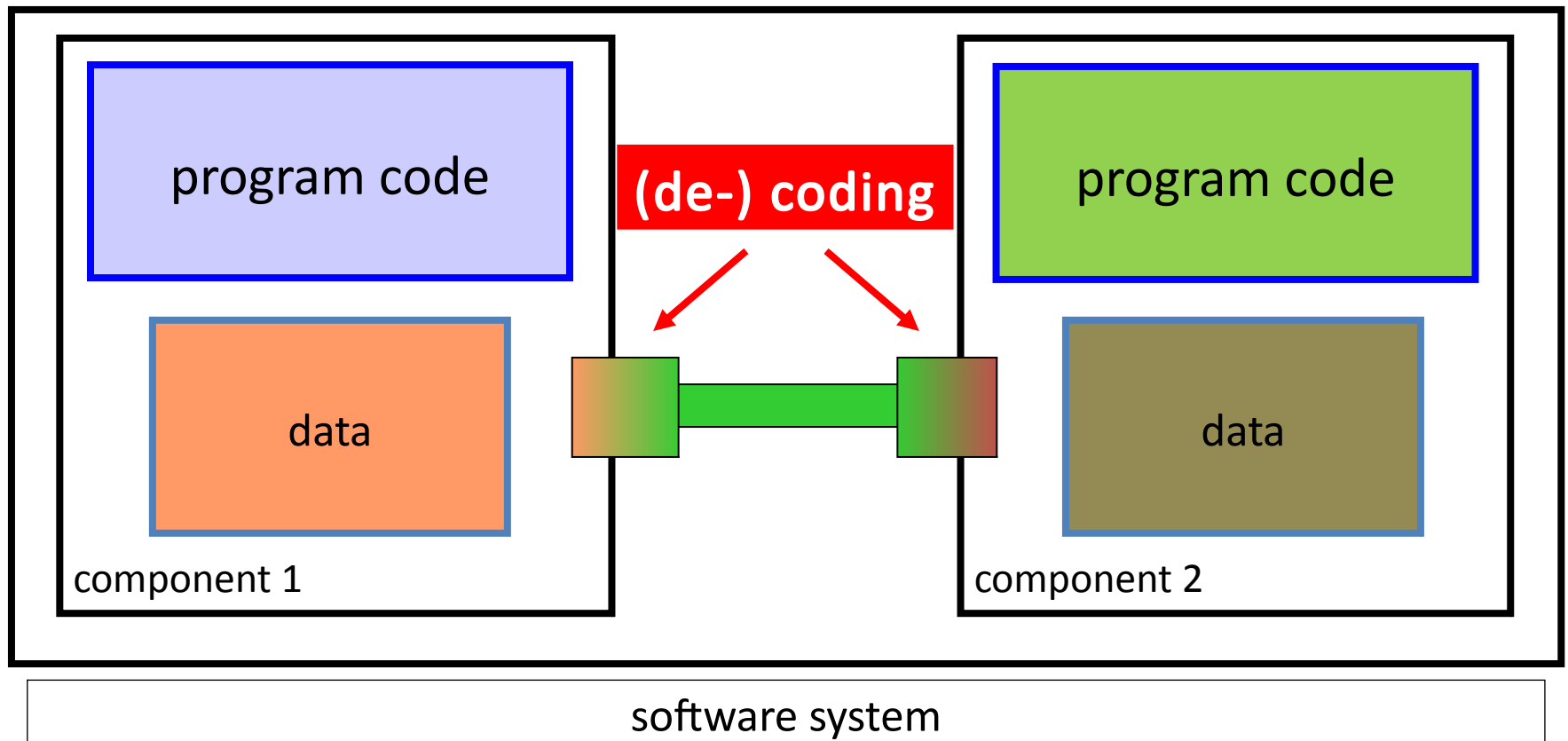
- Everything became bigger ...





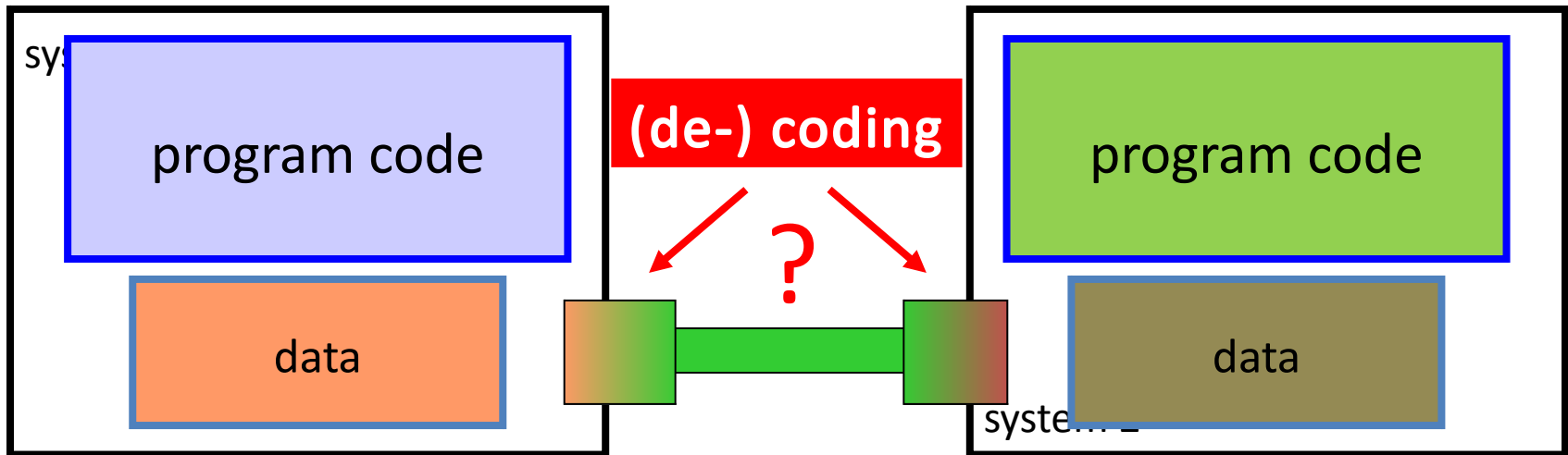
# XML in Communications

- ... and bigger



# XML in Communications

- ... and distributed

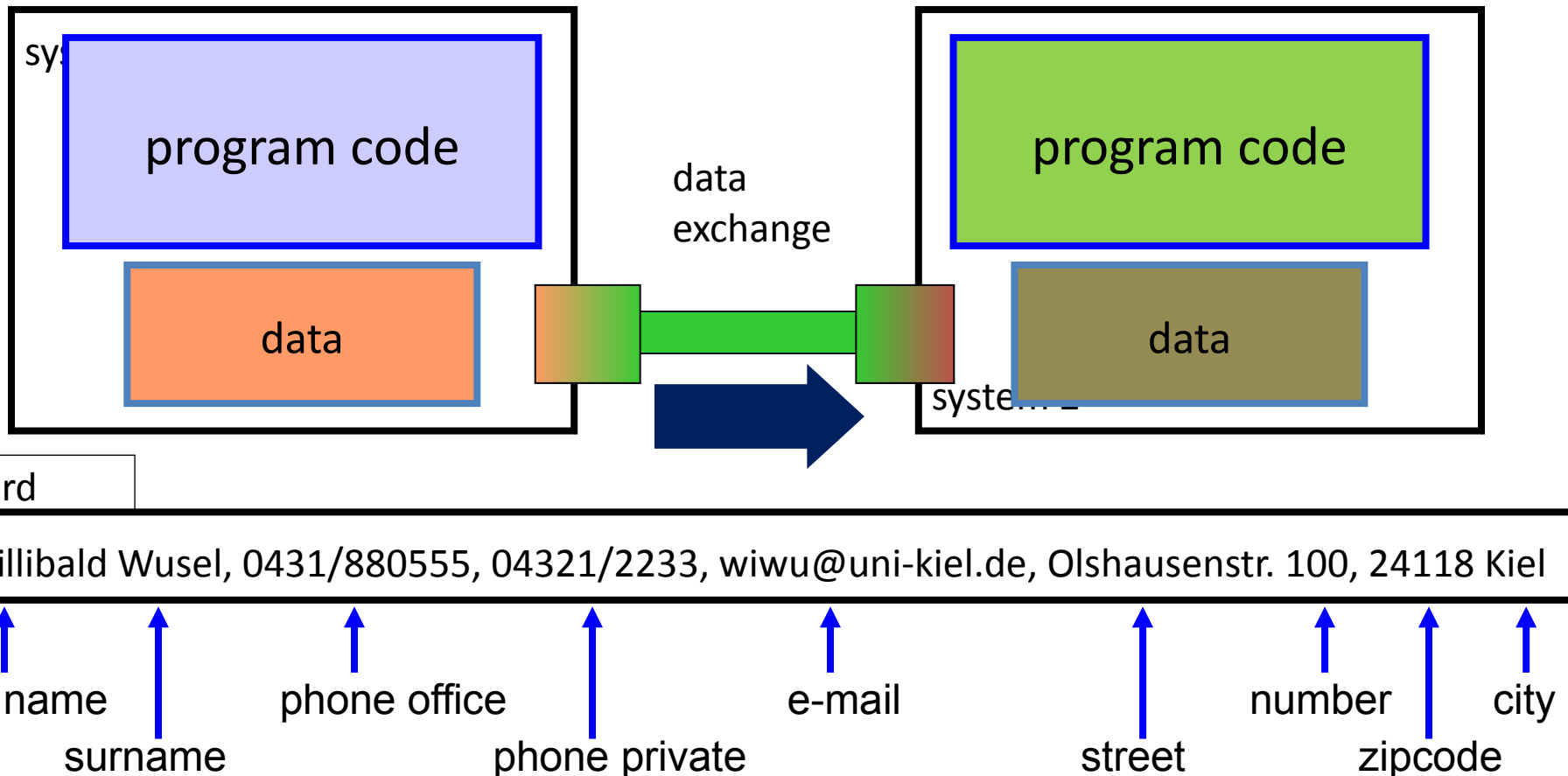


## Big questions:

- How to code data?
- How to manage data coding?

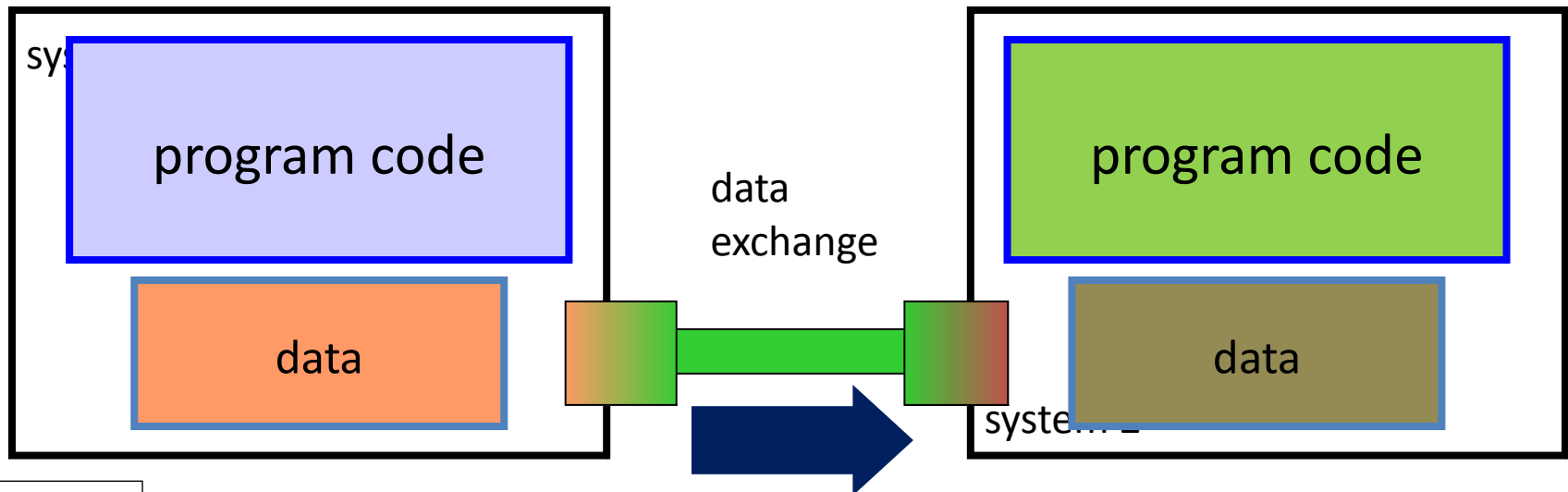
# XML in Communications

- Sample: record transmission



# XML in Communications

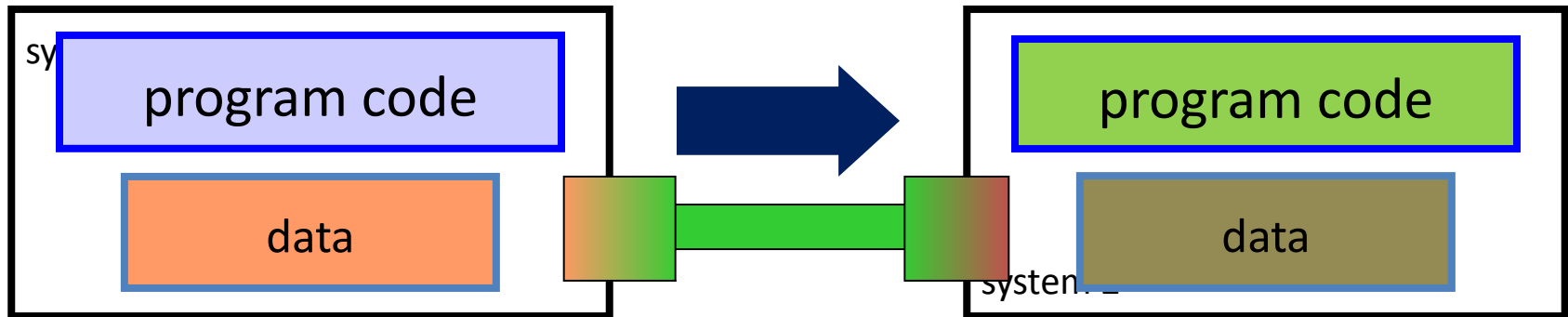
- Record: data + structure



record									
Willibald Wusel, 0431/880555, 04321/2233, wiwu@uni-kiel.de, Olshausenstr. 100, 24118 Kiel									
first name	↑	surname	↑	phone office	↑	phone private	↑	e-mail	↑
								street	↑
								number	↑
								zipcode	↑
									city

# XML in Communications

- Record: data + structure

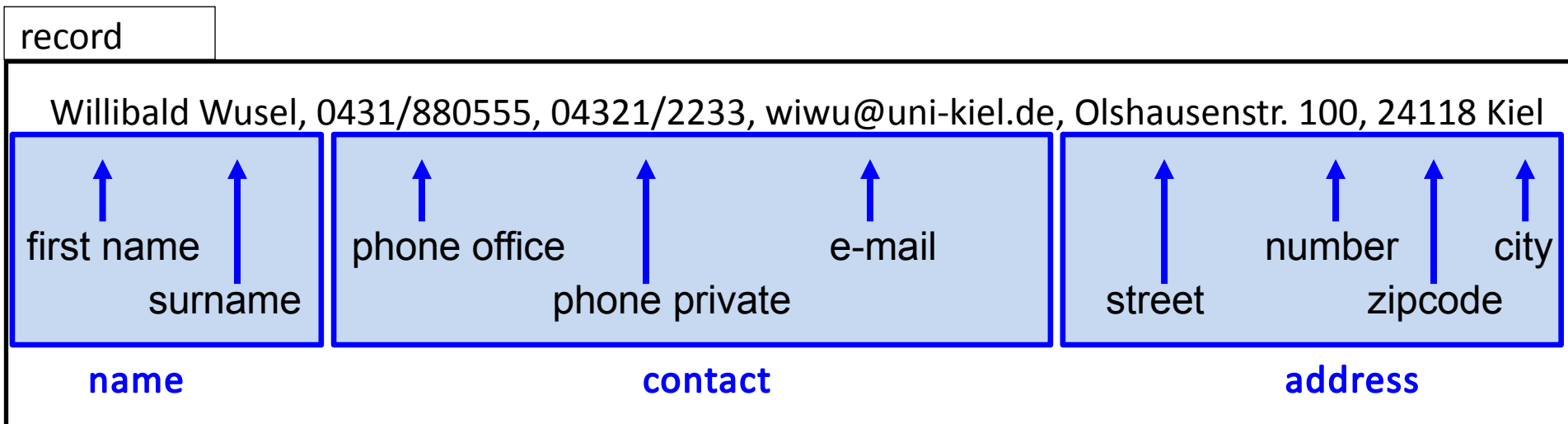
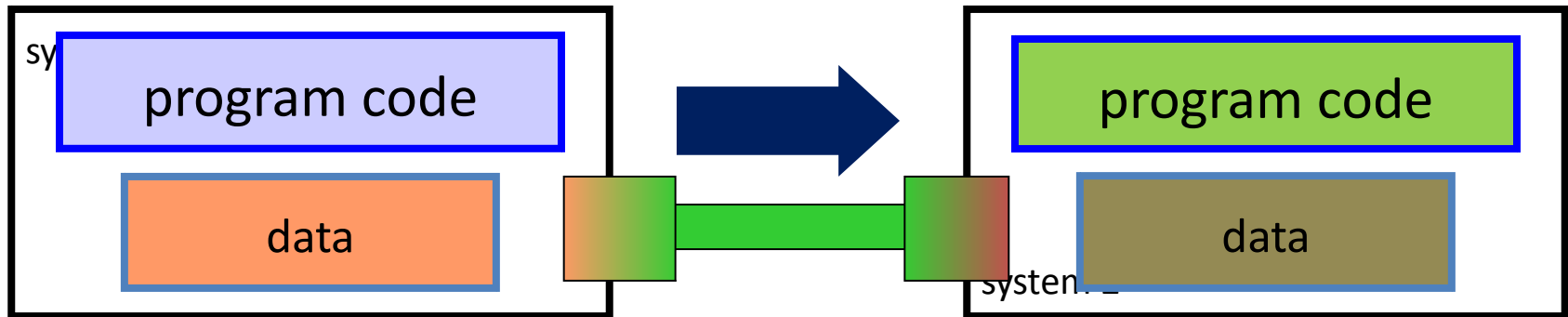


record

<first_name>	Willibald	</first_name>
<surname>	Wusel	</surname>
<phone_office>	0431/880555	</phone_office>
<phone_private>	04321/2233	</phone_private>
<e-mail>	wiwu@uni-kiel.de	</e-mail>
<street>	Olshausenstr.	</street>
<number>	100	</number>
<zipcode>	24118	</zipcode>
<city>	Kiel	</city>

# XML in Communications

- Nested structures

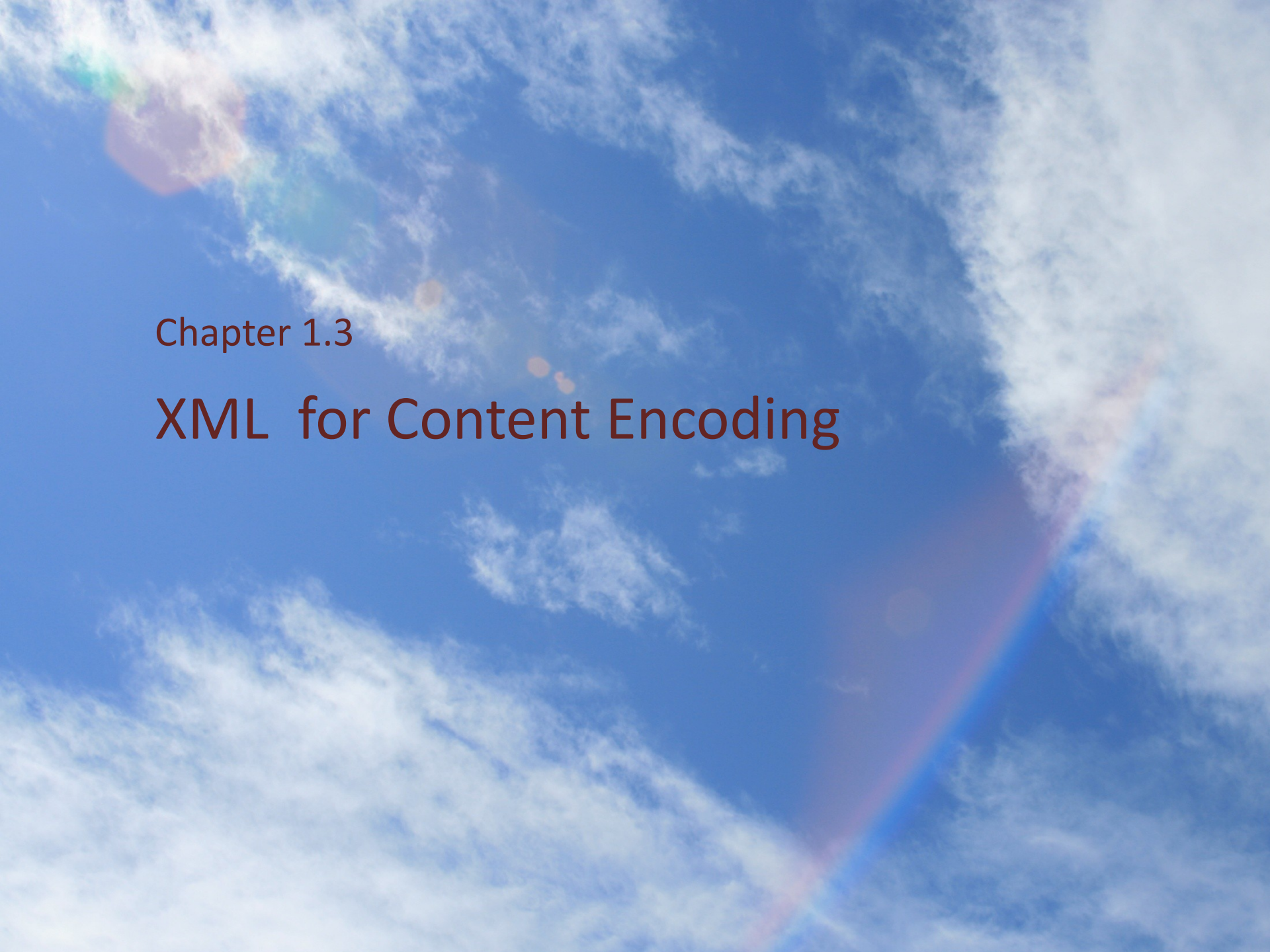


# XML in Communications

- Nested structures

record

```
<Name>
  <first_name>    Willibald    </first_name>
  <surname>       Wusel       </surname>
</Name>
<Contact>
  <phone_office>  0431/880555  </phone_office>
  <phone_private> 04321/2233   </phone_private>
  <e-mail>        wiwu@uni-kiel.de </e-mail>
</Contact>
<Address>
  <street>        Olshausenstr. </street>
  <number>        100          </number>
  <zipcode>       24118        </zipcode>
  <city>          Kiel         </city>
</Address>
```



Chapter 1.3

# XML for Content Encoding



# XML for Content Encoding

- Codd's idea of *logical-physical separation*
  - "The most important motivation for the research work ... was the objective of providing a sharp and clear boundary between the **logical and physical aspects** of database management."

E.F. Codd, Turing Award Lecture.

CACM 25 (2) (February 1982), 109-117

- The same *logical* data gets a different *physical* encoding, e.g.
  - converting from an in-memory data structure to something that can be sent elsewhere: "marshalling"
  - converting from an in-memory data structure to something that can be stored on non-volatile memory: "persistence"

# XML for Content Encoding

- Special problem: Metadata for locating documents
  - Text documents: Key word search is great, don't need metadata.
  - But other kinds of content?
    - spreadsheets
    - maps
    - purchase records
    - objects
    - images
    - ...

# XML for Content Encoding

- Metadata example: MP3 ID3 format – record at end of file

offset	length	description
0	3	"TAG" identifier string.
3	30	Song title string.
33	30	Artist string.
63	30	Album string.
93	4	Year string.
97	28	Comment string.
125	1	Zero byte separator.
126	1	Track byte.
127	1	Genre byte.

# XML for Content Encoding

- Metadata example: JPEG "JFIF" header
  - Start of Image (SOI) marker – two bytes (FFD8)
  - JFIF marker (FFE0)
  - length – two bytes
  - ASCII code for zero-terminated "JFIF" string – 4A-46-49-46-00
  - version – two bytes: often 01, 02
    - the most significant byte is used for major revisions
    - the least significant byte for minor revisions
  - units – one byte: units for the X and Y densities
    - 0 => no units, X and Y specify the pixel aspect ratio
    - 1 => X and Y are dots per inch
    - 2 => X and Y are dots per cm
  - $X_{\text{density}}$  – two bytes
  - $Y_{\text{density}}$  – two bytes
  - $X_{\text{thumbnail}}$  – one byte: 0 = no thumbnail
  - $Y_{\text{thumbnail}}$  – one byte: 0 = no thumbnail
  - (RGB)n – 3n bytes:  
packed (24-bit) RGB values for the thumbnail pixels,  $n = X_{\text{thumbnail}} * Y_{\text{thumbnail}}$

# XML for Content Encoding

- Desiderata for data interchange
  - Ability to represent many kinds of information represented by different data structures
  - Hardware-independent encoding: Endian-ness, UTF vs. ASCII vs. EBCDIC
  - Standard tools and interfaces
  - Ability to formally define grammar of expected data with forwards- and backwards-compatibility

→ That's XML ...

# A Few Common Uses of XML

- Storage format for DTP tools: "extensible HTML"
  - e.g. MS Office, OpenOffice
  - supplemented with stylesheets (XSL) to define typographic layout
- Exchange format for data
  - marshalling data in Web Services
  - need to agree on terminology → ontology
- Notation for modeling languages
  - kind of Domain-Specific Language (DSL)
  - example: Geography Markup Language (GML)

```

<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="4" name="Foliennummernplatzhalter 3"/>
    <p:cNvSpPr>
      <a:spLocks noGrp="1"/>
    </p:cNvSpPr>
    <p:nvPr>
      <p:ph type="sldNum" sz="quarter" idx="11"/>
    </p:nvPr>
  </p:nvSpPr>
<p:spPr/>
<p:txBody>
  <a:bodyPr/>
  <a:lstStyle/>
  <a:p>
    <a:fld id="{25589156-C03E-44C9-A3C4-DD8DDD26385A}" type="slidenum">|
      <a:rPr lang="de-DE" smtClean="0"/>
      <a:pPr/>
      <a:t>6</a:t>
    </a:fld>
    <a:endParaRPr lang="de-DE"/>
  </a:p>
</p:txBody>
</p:sp>
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="5" name="Inhaltsplatzhalter 4"/>
    <p:cNvSpPr>
      <a:spLocks noGrp="1"/>
    </p:cNvSpPr>
    <p:nvPr>
      <p:ph sz="quarter" idx="12"/>
    </p:nvPr>
  </p:nvSpPr>
<p:spPr/>
<p:txBody>
  <a:bodyPr/>
  <a:lstStyle/>
  <a:p>
    <a:r>
      <a:rPr lang="de-DE" smtClean="0"/>
      <a:t>Hello world!</a:t>
    </a:r>
    <a:endParaRPr lang="de-DE"/>
  </a:p>
</p:txBody>
</p:sp>

```

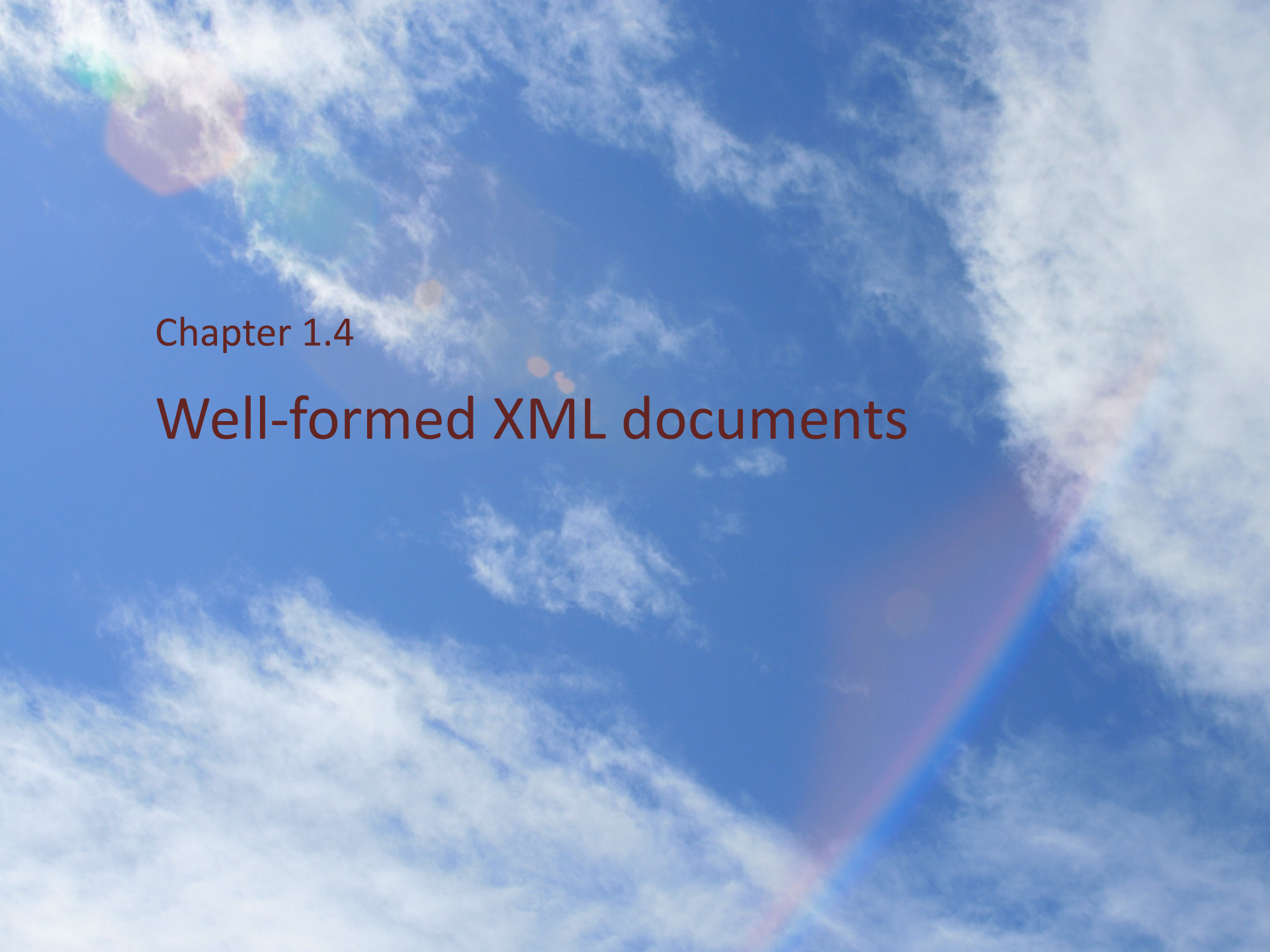
— Informatik · CAU Kiel

inside pptx  
(snippet)

# A Few Common Uses of XML

- Communities and business sectors are defining their specialized vocabularies
  - mathematics (MathML)
  - railway planning and operation (railML)
  - farm operation (agroML)
  - for further: see [http://en.wikipedia.org/wiki/List\\_of\\_XML\\_markup\\_languages](http://en.wikipedia.org/wiki/List_of_XML_markup_languages)





Chapter 1.4

# Well-formed XML documents

# Well-formed XML Documents

- A well-formed XML document consists of
  - a **prolog**
  - an **element**
  - an optional **epilog**
  - production:

`document ::= ( prolog element Misc* )`



# Well-formed XML Documents

- Prolog

`prolog ::= XMLDecl Misc* (doctypeddecl Misc*)?`

- The XML declaration consists of
  - Version info (required),
  - Encoding declaration (optional),
  - Standalone declaration (optional)

`<?xml`  `version="1.0"`  `encoding="UTF-8"`  `?>`



# Well-formed XML Documents

- Elements: the "things" the XML document talks about
  - E.g. books, authors, orders, invoices, tracks, signals, ...
- An element consists of:
  - an opening tag
  - the content
  - a closing tag

<lecturer>   Jesper Zedlitz   </lecturer>



# Well-formed XML Documents

- Elements (cont'd.)
  - Tag names are formed following production 5 of the XML spec,
  - and additionally must not begin with the string "xml" in any combination of cases.

for completeness:

[5] **Name** ::= NameStartChar (NameChar)\*

**NameStartChar** ::= ":" | [A-Z] | "\_" | [a-z] | [#xC0-#xD6] | [#xD8-#xF6] |  
[#xF8-#x2FF] | [#x370-#x37D] | [#x37F-#x1FFF] | [#x200C-#x200D] | [#x2070-#x218F] |  
[#x2C00-#x2FEF] | [#x3001-#xD7FF] | [#xF900-#xFDCF] | [#xFDF0-#xFFFD] |  
[#x10000-#xEFFFF]

**NameChar** ::= NameStartChar | "-" | "." | [0-9] | #xB7 | [#x0300-#x036F] |  
[#x203F-#x2040]

# Well-formed XML Documents

- Elements (cont'd.)
  - Element content may be text, or other elements, or a mixture thereof

```
<lecturer>Jesper Zedlitz</lecturer>
```

```
<lecturer>
```

```
  <name>Jesper Zedlitz</name>
```

```
</lecturer>
```

```
<lecturer>The lecturer of this lecture is
```

```
<name>Jesper Zedlitz</name> </lecturer>
```

- If there is no content, then the element is called empty; it is abbreviated as follows:

```
<lecturer/>  stands for  <lecturer></lecturer>
```



# Well-formed XML Documents

- Elements (cont'd.)
  - To attach **additional** information to an element, attributes can be used.
  - An attribute is a **name-value pair** separated by an equal sign (=).
  - An attribute is placed inside the **opening tag** of an element:

```
<lecturer name="Jesper Zedlitz"  
           phone="+49-431-880-7517" />
```



- An empty element is not necessarily meaningless:  
It may have some properties in terms of **attributes**.

# Well-formed XML Documents

- Example with/without attributes

```
<order orderNo="23456" customer="John Smith"
      date="October 15, 2002">
  <item itemNo="a528" quantity="1"/>
  <item itemNo="c817" quantity="3"/>
</order>
```

```
<order>
  <orderNo>23456</orderNo>
  <customer>John Smith</customer>
  <date>October 15, 2002</date>
  <item>
    <itemNo>a528</itemNo>
    <quantity>1</quantity> </item>
  <item>
    <itemNo>c817</itemNo>
    <quantity>3</quantity> </item>
</order>
```



# Well-formed XML Documents

- Attributes vs. elements
  - Attributes are part of markup, elements are part of the basic document contents.
  - Suggestion: use attributes for identifiers of elements, and use elements for content.
  - But: When to use elements and when attributes is a matter of taste.
  - Keep in mind:
    - Attributes **must not** be nested!
    - No fixed order of appearance for attributes!

# Well-formed XML Documents

- Further components of XML docs
  - Comments
    - A piece of text that is to be ignored by the parser



`<!-- This is a comment -->`

- Processing Instructions (PIs)

- Define procedural attachments

`<?stylesheet type="text/css" href="mystyle.css"?>`



- Both allowed in epilog
- More kinds of components to come

# Well-formed XML Documents

- Some syntactic rules for "well-formed" XML documents



One single outermost element.



Each element contains an opening and a corresponding closing tag.



Tags may not overlap.

`<author> <name> Lee Hong </name> </author>`



Element content may be characters, or other elements, or nothing.



Elements may have attributes given in the element start tag.



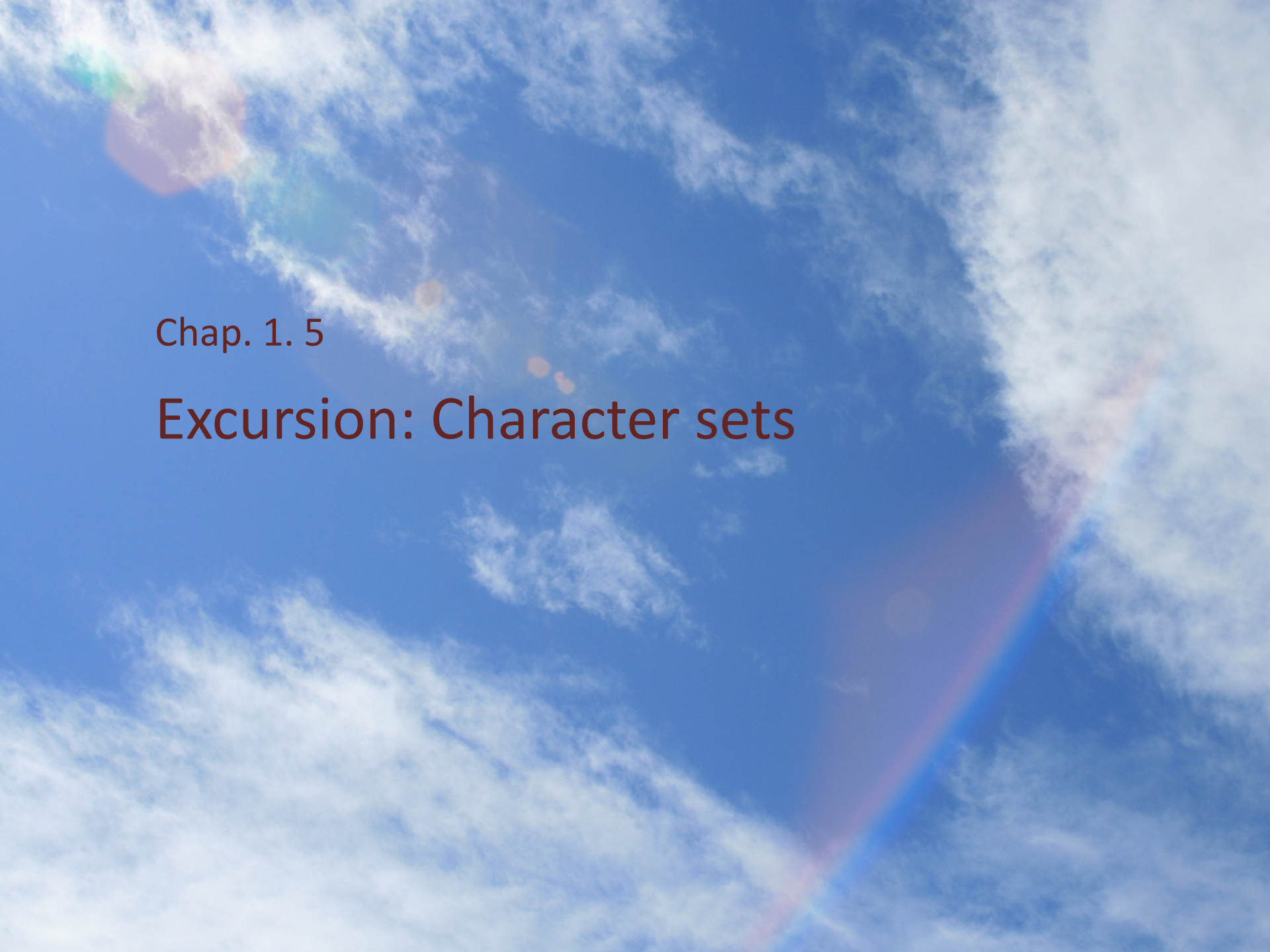
Attribute values must be put in parentheses.



XML is case-sensitive!



...



Chap. 1. 5

## Excursion: Character sets

# Character Sets

---

Computers don't understand text, they only understand numbers. For computers to be able to treat text, there must be a correspondence between numbers and text characters. Such a correspondence is called a coded character set.

Thomas Krichel

# Character Sets

- Important character sets
  - American Standard Code for Information Interchange (ASCII, US-ASCII)
  - ISO/IEC 646: International Reference Version (IRV); same as ASCII
  - ISO 8859 series of standards: 8-bit character encodings superseding the ISO 646 IRV and its national variants.
  - ISO 10646 standard: directly related to Unicode, superseding all of the ISO 646 and ISO 8859 sets of national-variant character encodings.

# Character Sets

- Terminology
  - **Character repertoire**: a set of distinct characters.
  - **Character code**: a mapping from a character repertoire to a set of non-negative integers, called
    - code points
    - code numbers
    - code values, or
    - code elements.
  - **Character encoding**: an algorithm for mapping code points into sequences of octets for storage or transmission.

# Character Sets

- Terminology example
  - ISO 10646

Repertoire	code point	16-bit encoding
LATIN SMALL LETTER A ("a")	U+0061 <sub>hex</sub>	00 61
EXCLAMATION MARK ("!")	U+0021 <sub>hex</sub>	00 21
LATIN SMALL LETTER A WITH DIAERESIS ("ä")	U+00E4 <sub>hex</sub>	00 E4
PER MILLE SIGN ("%")	U+2030 <sub>hex</sub>	20 30

- U+(four hex numbers) denote code points in ISO 10646.



# ASCII and IRV

- American Standard Code for Information Interchange
  - specifies 7-bit coding for space and 94 characters.
  - Character positions 0-31 and 127 are reserved for control codes.
  - Code 32 identifies a space.
  - Sequence:

! " # \$ % & ' ( ) \* + , - . / 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 { | } ~

# ASCII and IRV

- ASCII encoding

	0_	1_	2_	3_	4_	5_	6_	7_
_0	NUL	DLE	<sp>	0	@	P	'	p
_1	SOH	DC1	!	1	A	Q	a	q
_2	STX	DC2	"	2	B	R	b	r
_3	ETX	DC3	#	3	C	S	c	s
_4	EOT	DC4	\$	4	D	T	d	t
_5	ENQ	NAK	%	5	E	U	e	u
_6	ACK	SYN	&	6	F	V	f	v
_7	BEL	ETB	'	7	G	W	g	w
_8	BS	CAN	(	8	H	X	h	x
_9	HT	EM	)	9	I	Y	i	y
_A	LF	SUB	*	:	J	Z	j	z
_B	VT	ESC	+	;	K	[	k	{
_C	FF	FS	,	<	L	\	l	
_D	CR	GS	-	=	M	]	m	}
_E	SO	RS	.	>	N	^	n	~
_F	SI	US	/	?	O	_	o	DEL

# ISO 8859

- ISO 8859 extends ASCII with characters for western European languages
  - extending USASCII's 7-bit encoding to 8-bit encoding
  - positions 128 to 159 not used
  - default character set of html

# ISO 8859

- Overview

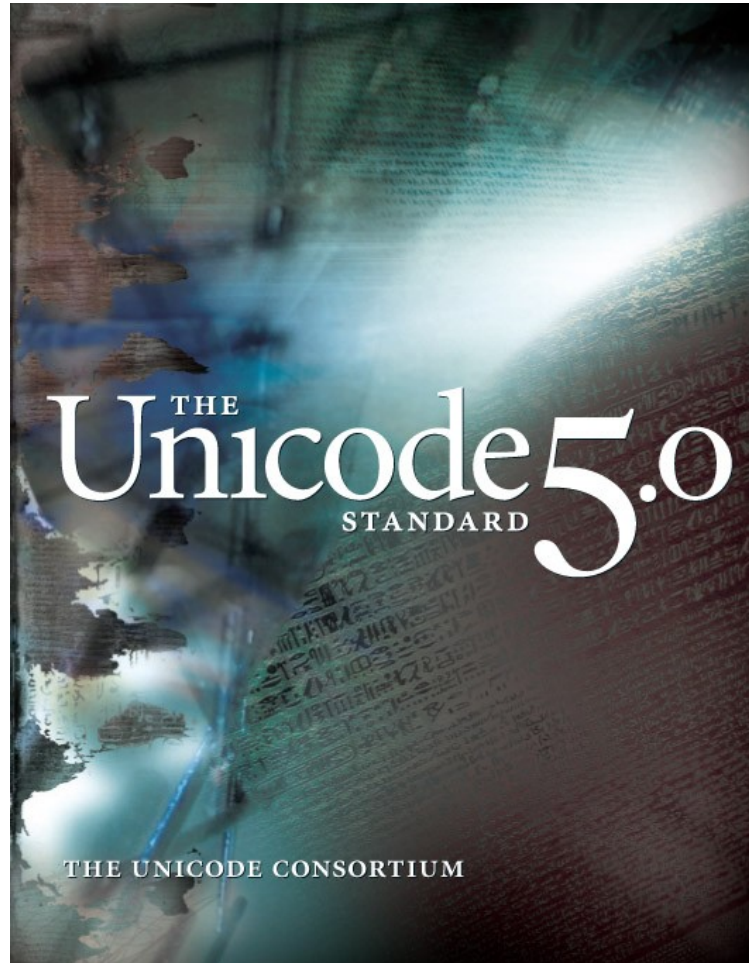
Part	1	Latin 1 (West European)	
	2	Latin 2 (East European)	Slavic, Albanian, Hungarian and a variation of Romanian
	3	Latin 3 (South European)	Southern European languages (Maltese) and Esperanto
	4	Latin 4 (North European)	Northern European languages
	5	Cyrillic	
	6	Arabic	
	7	Greek	under revision (to include, among others, the euro sign)
	8	Hebrew	
	9	Latin 5 (Turkish)	covers characters used for Turkish, replacing those in Part 1 for Icelandic
	10	Latin 6 (Nordic)	Icelandic, Nordic and Baltic character sets
	11	Latin/Thai	is being worked on
	12		
	13	Latin 7	
	14	Latin 8 (Celtic)	
	15	Latin 9	also supports the euro sign
	16	Latin 10	is being worked on to replace Part 2 for Romania and support some characters with comma below as opposed to with cedilla and also the euro sign

# ASCII and IRV

- National variants

Zeichenposition:	23	24	40	5B	5C	5D	5E	60	7B	7C	7D	7E
ISO 646-IRV	#	¤	@	[	\	]	^	`	{		}	~
Deutschland	#	\$	§	Ä	Ö	Ü	^	`	ä	ö	ü	ß
Schweiz	ù	\$	à	é	ç	ê	î	ô	ä	ö	ü	û
USA (ASCII)	#	\$	@	[	\	]	^	`	{		}	~
Großbritannien	£	\$	@	[	\	]	^	`	{		}	~
Frankreich	£	\$	à	°	ç	§	^	`	é	ù	è	”
Kanada	#	\$	à	â	ç	ê	î	ô	é	ù	è	û
Finnland	#	\$	@	Ä	Ö	Å	Ü	é	ä	ö	å	ü
Norwegen	#	\$	@	Æ	Ø	Å	^	`	æ	ø	å	~
Schweden	#	\$	É	Ä	Ö	Å	Ü	é	ä	ö	å	ü
Italien	£	\$	§	°	ç	é	^	ù	à	ò	ù	ì
Niederlande	£	\$	¾	ÿ	½		^	`	”	f	¼	’
Spanien	£	\$	§	í	Ñ	¿	^	`	°	ñ	ç	~
Portugal	#	\$	@	Ã	Ç	Õ	^	`	ã	ç	õ	~

# Unicode



# Unicode

- Features
  - The Unicode Standard, Version 5.0 provides codes for 1,114,112 characters from the world's alphabets, ideograph sets, and symbol collections.
  - The Unicode Standard further includes punctuation marks, diacritics, mathematical symbols, technical symbols, arrows, dingbats, etc.
  - ISO/IEC 10646 has been widely adopted in new Internet protocols and implemented in modern operating systems and computer languages.













from:  
[www.unicode.org/standard/principles.html](http://www.unicode.org/standard/principles.html)  
and ISO/IEC 10646

# What are "ideograph sets"?

(snippet from  
the Unicode  
standard)

## Ideographic description characters

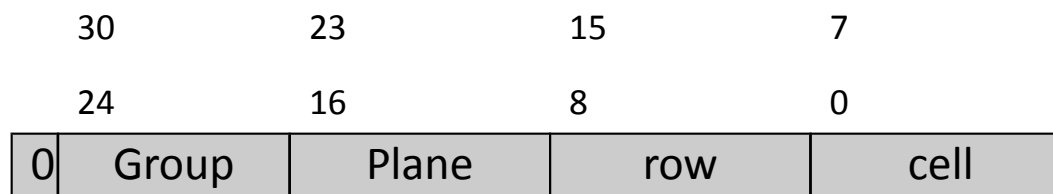
*These are visibly displayed graphic characters, not invisible composition controls.*

2FF0		IDEOGRAPHIC DESCRIPTION CHARACTER LEFT TO RIGHT
2FF1		IDEOGRAPHIC DESCRIPTION CHARACTER ABOVE TO BELOW
2FF2		IDEOGRAPHIC DESCRIPTION CHARACTER LEFT TO MIDDLE AND RIGHT
2FF3		IDEOGRAPHIC DESCRIPTION CHARACTER ABOVE TO MIDDLE AND BELOW
2FF4		IDEOGRAPHIC DESCRIPTION CHARACTER FULL SURROUND
2FF5		IDEOGRAPHIC DESCRIPTION CHARACTER SURROUND FROM ABOVE
2FF6		IDEOGRAPHIC DESCRIPTION CHARACTER SURROUND FROM BELOW
2FF7		IDEOGRAPHIC DESCRIPTION CHARACTER SURROUND FROM LEFT
2FF8		IDEOGRAPHIC DESCRIPTION CHARACTER SURROUND FROM UPPER LEFT
2FF9		IDEOGRAPHIC DESCRIPTION CHARACTER SURROUND FROM UPPER RIGHT
2FFA		IDEOGRAPHIC DESCRIPTION CHARACTER SURROUND FROM LOWER LEFT
2FFB		IDEOGRAPHIC DESCRIPTION CHARACTER OVERLAID



# Unicode

- Universal Character Set (UCS)
  - ISO 10646 formally defines a 31-bit character set.
  - Characters are represented as 32 bits, i.e. 4 bytes, or 8 hex chars (MSB = 0)
  - Four-dimensional coding space:
    - consisting of 128 groups
    - per group: 256 planes
    - per plane: 256 rows, each having 256 cells.

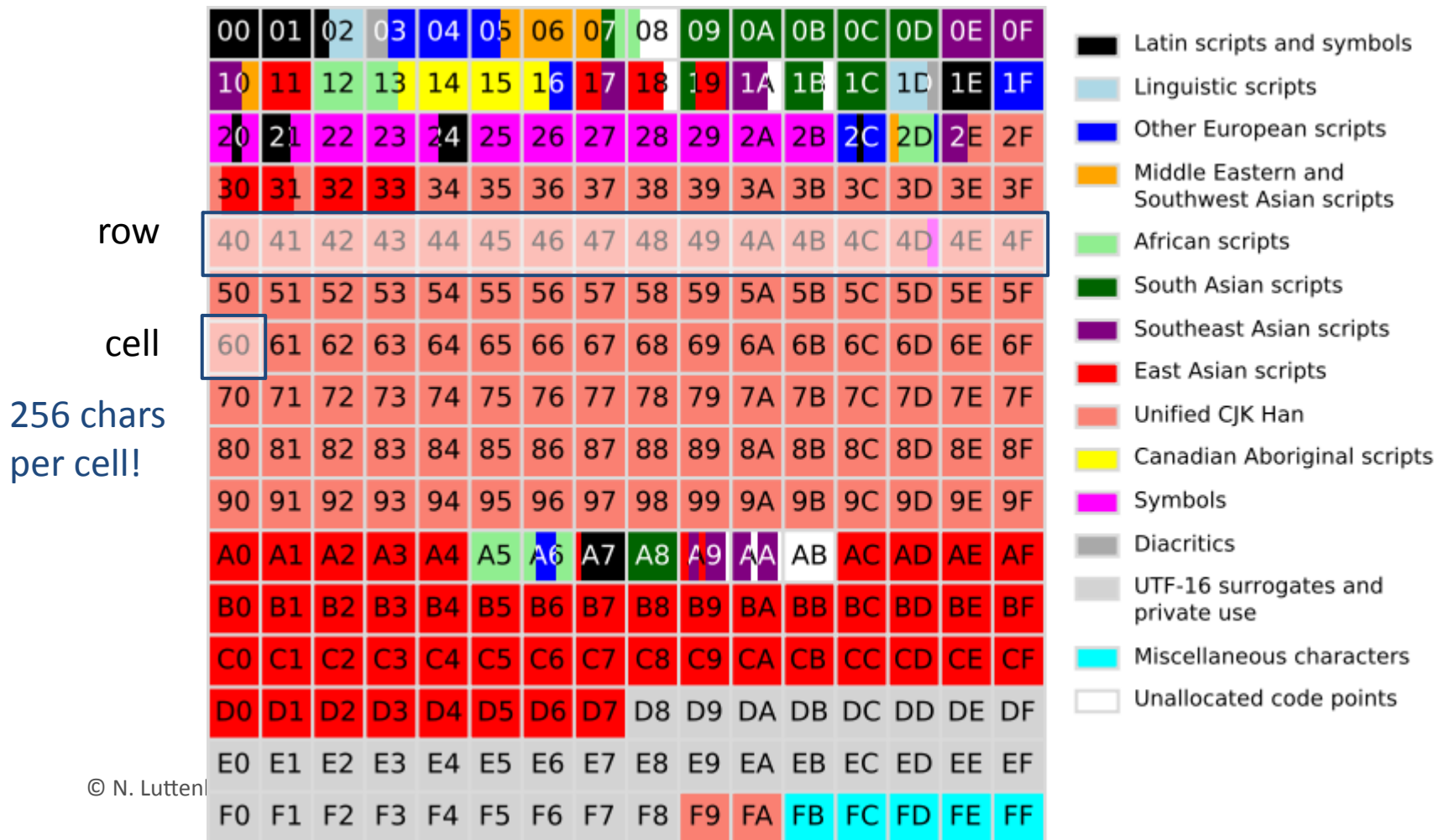


- Planes of the Universal Character Set (UCS)
  - **Plane 0**: Basic Multilingual Plane (BMP) contains the common-use characters for all the modern scripts of the world as well as many historical and rare characters.
  - **Plane 1**: Supplementary Multilingual Plane (SMP) is dedicated to the encoding of lesser-used historic scripts, special-purpose invented scripts, and special notational systems.
  - **Plane 2**: Supplementary Ideographic Plane (SIP) is intended as an additional allocation area for CJK characters.
  - **Plane 14**: Supplementary Special-purpose Plane (SSP) is the spillover allocation area for format control characters.
  - **Planes 15 and 16**: Private Use Planes. The two Private Use Planes are allocated, in their entirety, for private use.

- Planes of the Universal Character Set (UCS)
  - Group 0, Plane 0: [Basic Multilingual Plane](#), BMP
  - The 4-octet representation of a character in the BMP is produced by putting two 0 octets before its 2-octet representation.
  - The UCS characters U+0000 to U+007F are identical to those in ASCII.
  - The range U+0000 to U+00FF is identical to ISO 8859-1 (Latin-1).

# Unicode

- The Unicode Basic Multilingual Plane (BMP)



# Unicode


- Examples

Codepoint	Title	Block
Eintrag in:		
%	decode : [U+2030] PER MILLE SIGN	
黍	decode : [U+2FC9] KANGXI RADICAL MILLET	nt
%	decode : [U+0025] PERCENT SIGN	nt
秣	decode : [U+412E]	nt
秣	decode : [U+4130]	nt
秣	decode : [U+4136]	nt
秣	decode : [U+413E]	nt
秣	decode : [U+4142]	nt
粟	decode : [U+4147]	nt
秣	decode : [U+4155]	nt
秣	decode : [U+415F]	nt
秣	decode : [U+4163]	nt
秣	decode : [U+416D]	nt
糖	decode : [U+416F]	

# Unicode

- Encodings
  - UTF: Unicode Transformation Format
  - UTF-8
    - UCS characters U+0000 to U+007F (ASCII) are encoded as bytes 0x00 to 0x7F.
    - All other UCS characters are encoded as a sequence of bytes, each of which has the most significant bit set.
  - UTF-16
    - Coding of characters in the BMP; encoding equals code point

- Encodings
  - Byte order for UTF-16:
    - Unicode files start with the character U+FEFF (ZERO WIDTH NO-BREAK SPACE), also known as the [Byte-Order Mark](#) (BOM).
    - Big-endians deliver: FE FF
    - Little-endian deliver: FF FE,  
which is not a valid Unicode character
  - BOM for UTF-8: EF BB BF



Chap. 1.6

# An Initial »Shopping List«



# »Shopping List«

