ingenieur
wissenschaften
htw saar

Digital Skills für Ingenieur*innen

2. Vorlesungstag

„Datenformate“

ingenieur
wissenschaften
htw saar

Datenformate

Daten – das Gold des 21. Jahrhunderts

Mit dem Aufkommen von Social Media Plattformen, wie z.B.

- LinkedIn (2002)
- Myspace (2003)
- Facebook (2004) und
- Twitter (2007)

haben Internet-User*innen viele Ihrer eigenen Wünsche/Bedürfnisse preisgegeben. Diese Informationen sind Firmen bares Geld wert, da sie damit individualisierte Werbung schalten können.

ingenieur
wissenschaften
htw saar

Datenformate

Daten – das Gold des 21. Jahrhunderts

Data-driven Marketing

Mit dem Aufkommen von Social Media Plattformen, wie z.B.

- LinkedIn (2002)
- Myspace (2003)
- Facebook (2004) und
- Twitter (2007)

haben Internet-User*innen viele Ihrer eigenen Wünsche/Bedürfnisse preisgegeben. Diese Informationen sind Firmen bares Geld wert, da sie damit individualisierte Werbung schalten können.

ingenieur
wissenschaften
htw saar

Datenformate

Daten – das Gold des 21. Jahrhunderts

Mit dem Aufkommen von Social Media Plattformen, wie z.B.

- LinkedIn (2002)
- Myspace (2003)
- Facebook (2004) und
- Twitter (2007)

haben Internet-User*innen viele Ihrer eigenen Wünsche/Bedürfnisse preisgegeben. Diese Informationen sind Firmen bares Geld wert, da sie damit individualisierte Werbung schalten können.

Data-driven Marketing

- *Personalisierung des Kundenerlebnis, steigern der Kundentreue.*

Datenformate

Daten – das Gold des 21. Jahrhunderts

Mit dem Aufkommen von Social Media Plattformen, wie z.B.

- LinkedIn (2002)
- Myspace (2003)
- Facebook (2004) und
- Twitter (2007)

haben Internet-User*innen viele Ihrer eigenen Wünsche/Bedürfnisse preisgegeben. Diese Informationen sind Firmen bares Geld wert, da sie damit individualisierte Werbung schalten können.

Data-driven Marketing

- *Personalisierung des Kundenerlebnis, steigern der Kundentreue.*
- *Steigerung des Erfolgs von Werbekampagnen*

ingenieur
wissenschaften
htw saar

Datenformate

Daten – das Gold des 21. Jahrhunderts

Mit dem Aufkommen von Social Media Plattformen, wie z.B.

- LinkedIn (2002)
- Myspace (2003)
- Facebook (2004) und
- Twitter (2007)

haben Internet-User*innen viele Ihrer eigenen Wünsche/Bedürfnisse preisgegeben. Diese Informationen sind Firmen bares Geld wert, da sie damit individualisierte Werbung schalten können.

Data-driven Marketing

- *Personalisierung des Kundenerlebnis, steigern der Kundentreue.*
- *Steigerung des Erfolgs von Werbekampagnen*
- *Marketingleistung kann präzise bewertet werden. Identifikation von Risiko- und Wachstumsbereichen*

Datenformate

Daten – das Gold des 21. Jahrhunderts

Künftig wird allerdings KI unseren Alltag revolutionieren. Ein Beispiel hierfür ist ***IBM Watson for Oncology***:

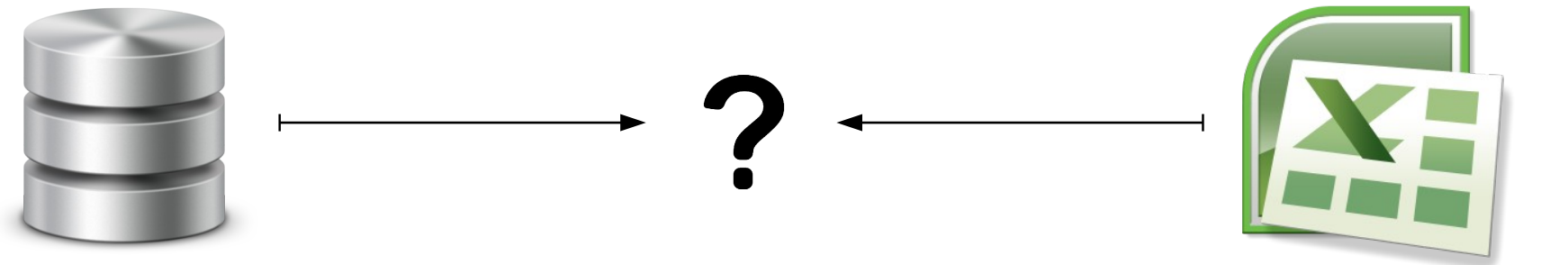
Künstliche Intelligenz – Plattform die weltweit ca. 230 Krankenhäuser bei onkologischen Fragestellungen Therapieoptionen vorschlägt.

→ Die Basis für das Training solcher Systeme sind Big Data. Diese Datenbasis muss in einer analysierbaren Form vorliegen.

Datenformate

Daten Austausch

Szenario: Austausch von Daten einer SQL Datenbank in ein Tabellenkalkulationsprogramm.



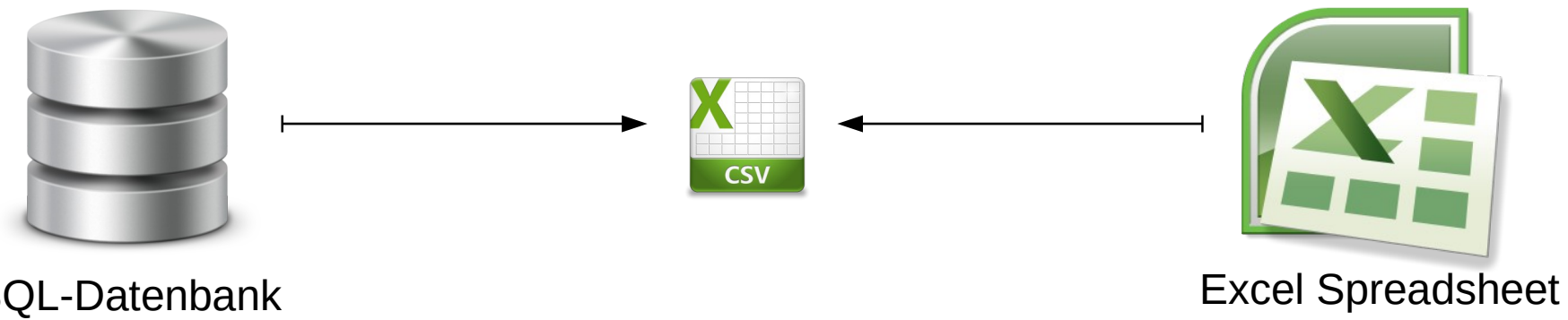
SQL-Datenbank

Excel Spreadsheet

Datenformate

Daten Austausch

Szenario: Austausch von Daten einer SQL Datenbank in ein Tabellenkalkulationsprogramm.



SQL-Datenbank

Excel Spreadsheet

Datenformate

Strukturierte Daten (z.B. CSV)

„The comma separated values format (CSV) has been used for exchanging and converting data between various spreadsheet programs for quite some time. Surprisingly, while this format is very common, it has never been formally documented.“ [2]

Bei einer CSV – Datei handelt es sich um strukturierte Daten. Denn es handelt sich dabei um Daten, die in einem vorgegebenen Format strukturiert werden, bevor sie im Datenspeicher abgelegt werden.

Innerhalb der RFC 4180 werden die nachfolgenden sieben Regeln für die Struktur formalisiert.

[1]: Vgl. https://datatracker.ietf.org/doc/html/rfc4180

Datenformate

Aufbau einer CSV – Datei

1. *„Jeder Datensatz befindet sich in einer separaten Zeile, die durch einen Zeilenumbruch (CRLF = Carriage Return Line Feed) getrennt ist.“* [1] Zum Beispiel:

aaa,bbb,ccc *[CRLF]*
zzz,yyy,xxx *[CRLF]*

[1]: Vgl. https://datatracker.ietf.org/doc/html/rfc4180

Datenformate

Aufbau einer CSV – Datei

2. *„Der letzte Datensatz in der Datei kann einen Zeilenumbruch am Ende haben oder auch nicht.“* [1] Zum Beispiel:

aaa,bbb,ccc *[CRLF]*
zzz,yyy,xxx

[1]: Vgl. https://datatracker.ietf.org/doc/html/rfc4180

Datenformate

Aufbau einer CSV – Datei

3. *„Möglicherweise erscheint als erste Zeile der Datei eine optionale Kopfzeile mit dem gleichen Format wie normale Datensatzzeilen. Dieser Header enthält Namen, die den Feldern in der Datei entsprechen, und sollte die gleiche Anzahl von Feldern enthalten wie die Datensätze im Rest der Datei (das Vorhandensein oder Fehlen der Header-Zeile sollte über den optionalen Parameter "header" dieser Datei angegeben werden Mime Typ).“* [1] Zum Beispiel:

feldname,feldname,feldname *[CRLF]*
aaa,bbb,ccc *[CRLF]*
zzz,yyy,xxx

[1]: Vgl. https://datatracker.ietf.org/doc/html/rfc4180

Datenformate

Aufbau einer CSV – Datei

4. *„Innerhalb des Headers und jedes Datensatzes können ein oder mehrere Felder vorhanden sein, die durch Kommas getrennt sind. Jede Zeile sollte in der gesamten Datei die gleiche Anzahl von Feldern enthalten. Leerzeichen werden als Teil eines Feldes betrachtet und sollten nicht ignoriert werden. Dem letzten Feld im Datensatz darf kein Komma folgen.“* [1] Zum Beispiel:

aaa,bbb,ccc

Anmerkung: Als Trennzeichen werden in der Praxis auch Semikolon, Tabulator, Leerzeichen, usw. genutzt.

[1]: Vgl. https://datatracker.ietf.org/doc/html/rfc4180

Datenformate

Aufbau einer CSV – Datei

5. *„Jedes Feld kann in doppelte Anführungszeichen gesetzt werden oder nicht (einige Programme, wie z. B. Microsoft Excel, verwenden jedoch überhaupt keine doppelten Anführungszeichen). Wenn Felder nicht in doppelte Anführungszeichen eingeschlossen sind, dürfen innerhalb der Felder keine doppelten Anführungszeichen erscheinen.“* [1]
Zum Beispiel:

"aaa","bbb","ccc" *[CRLF]*
zzz,yyy,xxx

[1]: Vgl. https://datatracker.ietf.org/doc/html/rfc4180

Datenformate

Aufbau einer CSV – Datei

6. *„Felder mit Zeilenumbrüchen (CRLF), doppelten Anführungszeichen und Kommas sollten in doppelte Anführungszeichen gesetzt werden.“* [1] Zum Beispiel:

"aaa","b *[CRLF]*
bb","ccc" *[CRLF]*
zzz,yyy,xxx

[1]: Vgl. https://datatracker.ietf.org/doc/html/rfc4180

Datenformate

Aufbau einer CSV – Datei

7. *„Wenn doppelte Anführungszeichen verwendet werden, um Felder zu umschließen, muss ein doppeltes Anführungszeichen, das innerhalb eines Felds erscheint, durch ein vorangestelltes doppeltes Anführungszeichen entwertet werden.“* [1] Zum Beispiel:

"aaa","b""bb","ccc"

[1]: Vgl. https://datatracker.ietf.org/doc/html/rfc4180

Datenformate

Vorname	Nachname	Geburtsdatum	Postleitzahl
Sieglinde	Müller	24.03.1928	66113
Josef	Franz	04.05.1948	66113
Peter	Jungblut	06.07.1963	66538

Tabelle 1: Personentabelle

Datenformate

Vorname	Nachname	Geburtsdatum	Postleitzahl
Sieglinde	Müller	24.03.1928	66113
Josef	Franz	04.05.1948	66113
Peter	Jungblut	06.07.1963	66538

Tabelle 1: Personentabelle

```
Vorname;Nachname;Geburtsdatum;Postleitzahl
"Sieglinde";"Müller";"24.03.1928";"66113"
"Josef";"Franz";"04.05.1948";"66113"
"Peter";"Jungblut";"06.07.1963";"66538"
```

Personen.csv

Datenformate

Vorteile strukturierter Daten

Vorteile

- Einfach nutzbar für Machine Learning Algorithmen. Da die Daten strukturiert sind, können Sie leicht bearbeitet und abgefragt werden.
- Ein*e durchschnittliche*r Benutzer*in, die/der sich mit dem Thema auskennt, auf das sich die Daten beziehen, kann diese auch nutzen (und auch lesen).
- Da strukturierte Daten weit verbreitet sind, unterstützen viele Tools Import-/Export-Möglichkeiten.

Datenformate

Nachteile strukturierter Daten

Nachteile

- Ein vordefinierter Zweck schränkt die Verwendungsmöglichkeiten ein. Somit haben strukturierte Daten eine geringere Flexibilität und Anwendungsmöglichkeit.

- Das Speichern strukturierter Daten folgt starren Schemata. Jede Anforderungsänderung bedeutet: alle strukturierten Daten müssen aktualisiert werden, damit sie den neuen Anforderungen entsprechen. Dadurch entsteht ein größerer Aufwand von Zeit und Ressourcen.

Datenformate

Semistrukturierte Daten (z.B. JSON)

JSON

*„**JavaScript Object Notation** (JSON) ist ein einfaches, textbasiertes, sprachunabhängiges Datenaustauschformat. Es wurde vom ECMAScript Programming Language Standard abgeleitet. JSON definiert einen kleinen Satz von Formatierungsregeln für die portable Darstellung strukturierter Daten.“* [2] [3]

JSON wird durch die beiden inhaltsgleichen Standards RFC 8259 und ECMA-404 definiert.

[2]: Vgl. https://datatracker.ietf.org/doc/html/rfc8259
[3]: Vgl. https://www.ecma-international.org/publications-and-standards/standards/ecma-404/

Datenformate

JSON

JSON Grammatik (1)

„Ein JSON-Text ist eine Folge von Tokens. Der Token-Satz umfasst sechs Strukturzeichen, Zeichenfolgen, Zahlen und drei wörtliche Namen.“ [2]

Strukturzeichen

1. begin-array	= [	2. end-array	=]
3. begin-object	= {	4. end-object	= }
5. name-seperator	= :	6. value-seperator	= ,

[2]: Vgl. https://datatracker.ietf.org/doc/html/rfc8259

Datenformate

JSON

JSON Grammatik (2)

„*Unbedeutende Leerzeichen sind vor oder nach jedem der sechs Strukturzeichen erlaubt.*“ [2]

Unbedeutende Leerzeichen sind definiert als Space (%x20), Horizontal Tab (%x09), Line feed oder New line (%x0A) und Carriage Return (%x0D).

[2]: Vgl. https://datatracker.ietf.org/doc/html/rfc8259

Datenformate

JSON

Werte

„Ein JSON-Wert muss ein Objekt, ein Array, eine Zahl oder eine Zeichenfolge oder einer der folgenden drei wörtliche Namen sein: false, null, true*. Die wörtlichen Namen müssen in Kleinbuchstaben geschrieben werden. Andere wörtliche Namen sind nicht zulässig.“* [2]

value = false / null / true / object / array / number / string

[2]: Vgl. https://datatracker.ietf.org/doc/html/rfc8259

Datenformate

JSON

Objekte (1)

„Eine Objektstruktur wird als Paar geschweifter Klammern dargestellt, die null oder mehr Attribut-Wert-Paare (oder Mitglieder) umgeben. Ein Attribut ist eine Zeichenfolge. Nach jedem Attribut steht ein einzelner Doppelpunkt, der das Attribut vom Wert trennt. Ein einzelnes Komma trennt einen Wert von einem nachfolgenden Attribut. Die Attribute innerhalb eines Objekts sollten eindeutig sein.“ [2]

[2]: Vgl. https://datatracker.ietf.org/doc/html/rfc8259

Datenformate

JSON

Objekte (2)

„Ein Objekt, dessen Attribute alle eindeutig sind, ist in dem Sinne interoperabel, dass alle Softwareimplementierungen, die dieses Objekt empfangen, sich auf die Attribut-Wert-Zuordnungen einigen werden. Wenn die Attribute innerhalb eines Objekts nicht eindeutig sind, ist das Verhalten von Software, die ein solches Objekt empfängt, unvorhersehbar. Viele Implementierungen melden nur das letzte Attribut-Wert-Paar.“ [2]

[2]: Vgl. https://datatracker.ietf.org/doc/html/rfc8259

Datenformate

JSON

Arrays

„Eine Array-Struktur wird als eckige Klammern dargestellt, die null oder mehr Werte (oder Elemente) umgeben. Elemente werden durch Kommas getrennt. Es ist nicht erforderlich, dass die Werte in einem Array denselben Typ haben.“ [2]

[2]: Vgl. https://datatracker.ietf.org/doc/html/rfc8259

Datenformate

JSON

Zahlen (1)

„Die Darstellung von Zahlen ähnelt der in den meisten Programmiersprachen verwendeten. Eine Zahl wird zur Basis 10 mit Dezimalziffern dargestellt. Es enthält eine ganzzahlige Komponente, der optional ein Minuszeichen vorangestellt werden kann, gefolgt von einem Bruchteil und/oder einem Exponententeil. Führende Nullen sind nicht erlaubt.“ [2]

number = [minus] int [frac] [exp]

[2]: Vgl. https://datatracker.ietf.org/doc/html/rfc8259

Datenformate

JSON

Zahlen (2)

„Ein Bruchteil ist ein Dezimalpunkt, gefolgt von einer oder mehreren Ziffern. Ein Exponent beginnt mit dem Buchstaben E in Groß- oder Kleinbuchstaben, dem ein Plus- oder Minuszeichen folgen kann. Auf das E und das optionale Zeichen folgen eine oder mehrere Ziffern.“ [2]

decimal-point = . digit1-9 = 1-9 e = e E
exp = e [minus/plus]1*DIGIT
frac = decimal-point 1*DIGIT

[2]: Vgl. https://datatracker.ietf.org/doc/html/rfc8259

Datenformate

JSON

Strings

*„Die Darstellung von Strings ähnelt den Konventionen, die in der C-Familie von Programmiersprachen verwendet werden. Ein String beginnt und endet mit Anführungszeichen. Alle Unicode-Zeichen können innerhalb der Anführungszeichen gesetzt werden, mit Ausnahme der Zeichen, die entwertet werden müssen: Anführungszeichen, Backslash und die Steuerzeichen (*U+0000 *bis* U+001F*).“* [2]

Anmerkung: Entwertet wird kompakt mit "\" oder mit "\u+Unicode Code Point", z.B. kann der String "\" als "\u005C" oder kurz als "\\" repräsentiert werden.

[2]: Vgl. https://datatracker.ietf.org/doc/html/rfc8259

ingenieur
wissenschaften
htw saar

Datenformate

JSON

Beispiel

```
{
  "Person":{
    "name":"John Smith",
    "isAlive":true,
    "age":25,
    "address":{
      "cityStreet":"New York, 21 2nd Street",
      "postalCode":"10021-31\\00"
    },
    "children":[
      "peter",
      12,
      false
    ],
    "spouse":null
  }
}
```

[3]

[3]: Vgl. https://jsonformatter.curiousconcept.com/

Datenformate

Semistrukturierte Daten (z.B. XML)

XML

„***Extensible Markup Language***, *abgekürzt XML, beschreibt eine Klasse von Datenobjekten, die als XML-Dokumente bezeichnet werden, und beschreibt teilweise das Verhalten von Computerprogrammen, die sie verarbeiten. XML ist ein Anwendungsprofil oder eine eingeschränkte Form von SGML, der* ***Standard Generalized Markup Language****. XML-Dokumente sind konstruktionsbedingt konforme SGML-Dokumente.*“

[4]: Vgl. https://www.w3.org/TR/REC-xml/

Datenformate

XML

„XML ist eine Auszeichnungssprache zum Strukturieren von Dokumenten und Daten. Auszeichnungssprachen (engl. Markup Languages) geben Regeln vor, mit denen Teile eines Textes in bestimmter Weise markiert (ausgezeichnet) werden können, um ihnen zusätzliche Semantik und Eigenschaften hinzuzufügen, meist mit dem Ziel einen Text maschinenlesbar zu machen.“

[4]: Vgl. https://www.w3.org/TR/REC-xml/

Datenformate

XML

Beispiel

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<verzeichnis>
    <titel>Wikipedia Städteverzeichnis</titel>
    <eintrag>
        <stichwort>Genf</stichwort>
        <eintragstext>Genf ist der Sitz von ...</eintragstext>
    </eintrag>
    <eintrag>
        <stichwort>Köln</stichwort>
        <eintragstext>Köln ist eine Stadt, die ...</eintragstext>
    </eintrag>
</verzeichnis>
```

[5]

[5]: Vgl. https://de.wikipedia.org/wiki/Extensible_Markup_Language

Datenformate

XML

Dokumententität : Dies ist die Haupt-XML-Datei.

Beispiel

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<verzeichnis>
    <titel>Wikipedia Städteverzeichnis</titel>
    <eintrag>
        <stichwort>Genf</stichwort>
        <eintragstext>Genf ist der Sitz von ...</eintragstext>
    </eintrag>
    <eintrag>
        <stichwort>Köln</stichwort>
        <eintragstext>Köln ist eine Stadt, die ...</eintragstext>
    </eintrag>
</verzeichnis>
```

[5]

[5]: Vgl. https://de.wikipedia.org/wiki/Extensible_Markup_Language

ingenieur
wissenschaften
htw saar

Datenformate

XML

XML-Deklaration enthält:
- Version
- Zeichenkodierung
- Verarbeitbarkeit ohne Document Type Definition

Beispiel

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<verzeichnis>
    <titel>Wikipedia Städteverzeichnis</titel>
    <eintrag>
        <stichwort>Genf</stichwort>
        <eintragstext>Genf ist der Sitz von ...</eintragstext>
    </eintrag>
    <eintrag>
        <stichwort>Köln</stichwort>
        <eintragstext>Köln ist eine Stadt, die ...</eintragstext>
    </eintrag>
</verzeichnis>
```

[5]

[5]: Vgl. https://de.wikipedia.org/wiki/Extensible_Markup_Language

ingenieur
wissenschaften
htw saar

Datenformate

XML

Das Dokument besitzt genau ein Wurzelelement. Als Wurzelelement wird dabei das jeweils äußerste Element bezeichnet, z.B. ist in diesem Fall das Element verzeichnis Wurzelelement (Root-Element).

Beispiel

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<verzeichnis>
    <titel>Wikipedia Städteverzeichnis</titel>
    <eintrag>
        <stichwort>Genf</stichwort>
        <eintragstext>Genf ist der Sitz von ...</eintragstext>
    </eintrag>
    <eintrag>
        <stichwort>Köln</stichwort>
        <eintragstext>Köln ist eine Stadt, die ...</eintragstext>
    </eintrag>
</verzeichnis>
```

[5]

[5]: Vgl. https://de.wikipedia.org/wiki/Extensible_Markup_Language

ingenieur
wissenschaften
htw saar

Datenformate

XML

Starttag für den Beginn des Wurzelelement verzeichnis.

Beispiel

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<verzeichnis>
    <titel>Wikipedia Städteverzeichnis</titel>
    <eintrag>
        <stichwort>Genf</stichwort>
        <eintragstext>Genf ist der Sitz von ...</eintragstext>
    </eintrag>
    <eintrag>
        <stichwort>Köln</stichwort>
        <eintragstext>Köln ist eine Stadt, die ...</eintragstext>
    </eintrag>
</verzeichnis>
```

[5]

[5]: Vgl. https://de.wikipedia.org/wiki/Extensible_Markup_Language

ingenieur wissenschaften htw saar

Datenformate

XML

Endtag für das Ende des Wurzelelement verzeichnis.

Beispiel

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<verzeichnis>
    <titel>Wikipedia Städteverzeichnis</titel>
    <eintrag>
        <stichwort>Genf</stichwort>
        <eintragstext>Genf ist der Sitz von ...</eintragstext>
    </eintrag>
    <eintrag>
        <stichwort>Köln</stichwort>
        <eintragstext>Köln ist eine Stadt, die ...</eintragstext>
    </eintrag>
</verzeichnis>
```

[5]

[5]: Vgl. https://de.wikipedia.org/wiki/Extensible_Markup_Language

Datenformate

XML

Elemente ohne Inhalt können mit einem Leertag gekennzeichnet werden, z.B. <leertag/>.

Beispiel

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<verzeichnis>
    <titel>Wikipedia Städteverzeichnis</titel>
    <eintrag>
        <stichwort>Genf</stichwort>
        <eintragstext>Genf ist der Sitz von ...</eintragstext>
    </eintrag>
    <eintrag>
        <stichwort>Köln</stichwort>
        <eintragstext>Köln ist eine Stadt, die ...</eintragstext>
    </eintrag>
</verzeichnis>
```

[5]

[5]: Vgl. https://de.wikipedia.org/wiki/Extensible_Markup_Language

Datenformate

XML

Wohlgeformtheit (well-formed)

„*Ein XML-Dokument heißt „wohlgeformt“ (oder englisch well-formed), wenn es alle XML-Regeln einhält.*“ [5]

- Das Dokument besitzt genau ein Wurzelelement. Als Wurzelelement wird dabei das jeweils äußerste Element bezeichnet.

- Alle Elemente mit Inhalt besitzen einen Start- und einen Endtag (z. B. <eintrag>Eintrag 1</eintrag>). Elemente ohne Inhalt können mit einem Leertag gekennzeichnet werden (z. B. <eintrag />).

[5]: Vgl. https://de.wikipedia.org/wiki/Extensible_Markup_Language

Datenformate

XML

Wohlgeformtheit (well-formed)

- Die Start- und Endtags sind ebenentreu-paarig verschachtelt. Das bedeutet, dass alle Elemente geschlossen werden müssen, bevor die End-Auszeichner des entsprechenden Elternelements oder die Beginn-Auszeichner eines Geschwisterelements erscheinen.
- Ein Element darf nicht mehrere Attribute mit demselben Namen besitzen.

[5]: Vgl. https://de.wikipedia.org/wiki/Extensible_Markup_Language

Datenformate

XML

Wohlgeformtheit (well-formed)

- Attributwerte müssen in Anführungszeichen stehen ("..." oder '...').
- Die Start- und Endtags beachten die Groß- und Kleinschreibung (z. B. <eintrag></Eintrag> ist nicht gültig).

[5]: Vgl. https://de.wikipedia.org/wiki/Extensible_Markup_Language

ingenieur
wissenschaften
htw saar

Datenformate

Fazit

CSV		JSON		XML	
- geringe Flexibilität	+ einfach zu implementieren und zu parsen	- ist keine so robuste Datenstruktur wie XML	+ ist in seiner erweiterten Form einfacher zu lesen als XML	- sehr ausführlich und redundant.	+ XML ermöglicht die Validierung mit DTD und Schema
- nicht geeignet für verschachtelte Strukturen	+ kompakt im Gegensatz zu XML, da keine Tags verwendet werden	- ist nicht gut Informationen aus verschiedenen Systemen zu kombinieren	+ geringere Zeichenanzahl → geringere Overhead bei Datentransfer	- weniger übersichtlich als JSON	+ Plattform und Programmiersprachen unabhängig
- keine Unterscheidung zwischen Zahlen und Text	+ leicht zu generieren	- Fehler sind teilweise schwerer festzustellen als in XML	+ einfacher zu parsen	- unterstützt keine Arrays.	+ Unicode → Transport jeder menschlichen Sprache möglich

Literatur

[1]: Vgl. https://datatracker.ietf.org/doc/html/rfc4180
[2]: Vgl. https://datatracker.ietf.org/doc/html/rfc8259
[3]: Vgl. https://www.ecma-international.org/publications-and-standards/standards/ecma-404/
[4]: Vgl. https://www.w3.org/TR/REC-xml/
[5]: Vgl. https://de.wikipedia.org/wiki/Extensible_Markup_Language