XML

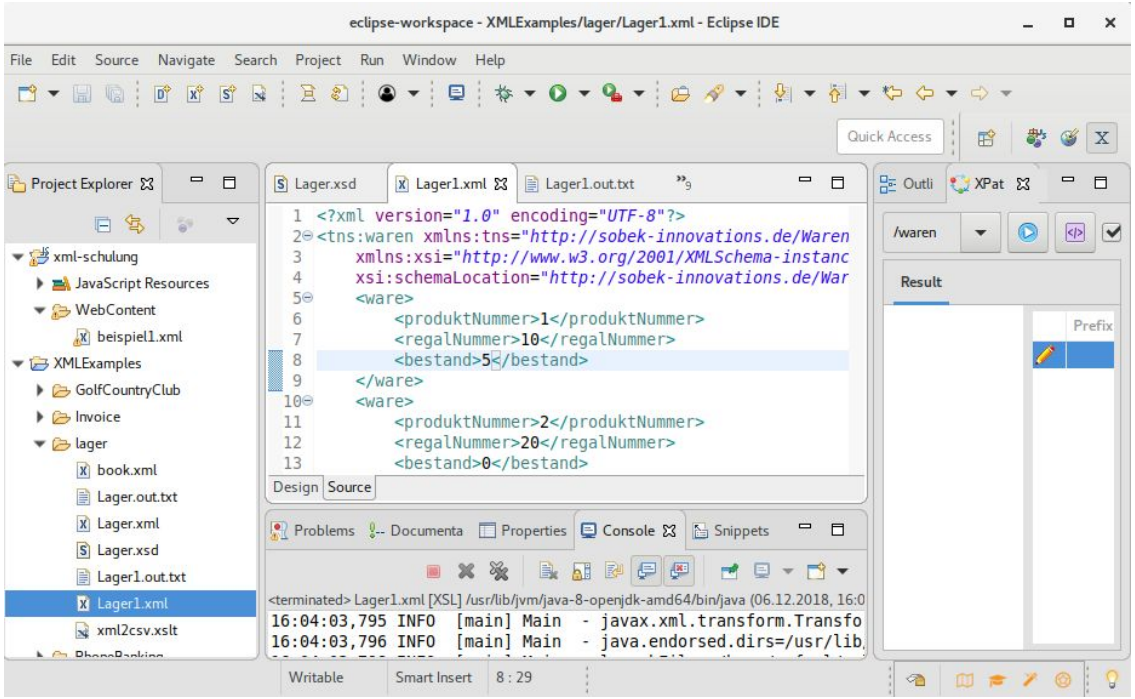
Vorstellung

Vorstellung Teilnehmer

Was ist eure Erwartungshaltung?
Erfahrung mit XML?
Erfahrung mit Schemas u. XSLT?

Vorbereitung

IDE-Installation // Eclipse



https://www.eclipse.org/downloads/packages/release/2018-09/r/eclipse-ide-javascript-and-web-developers

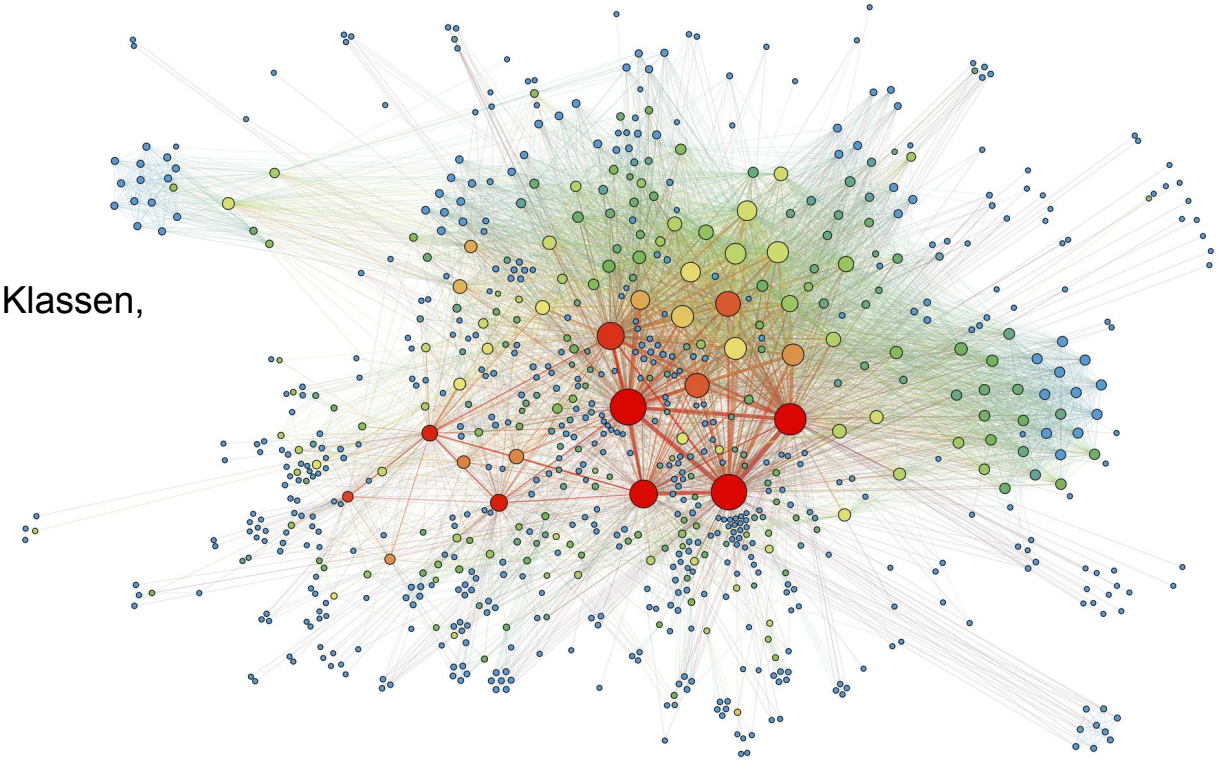
IDE-Installation // XPath Plugin + XSLT WTP

XPath Plugin Repo in Available Software eintragen und installieren -
https://raw.githubusercontent.com/stoupa91/eclipse-xpath-evaluation-plugin/master/eclipse-xpath-evaluation-plugin-update-site/

XML - Der Mensch und sein Umgang mit Komplexität I

aus Wikipedia

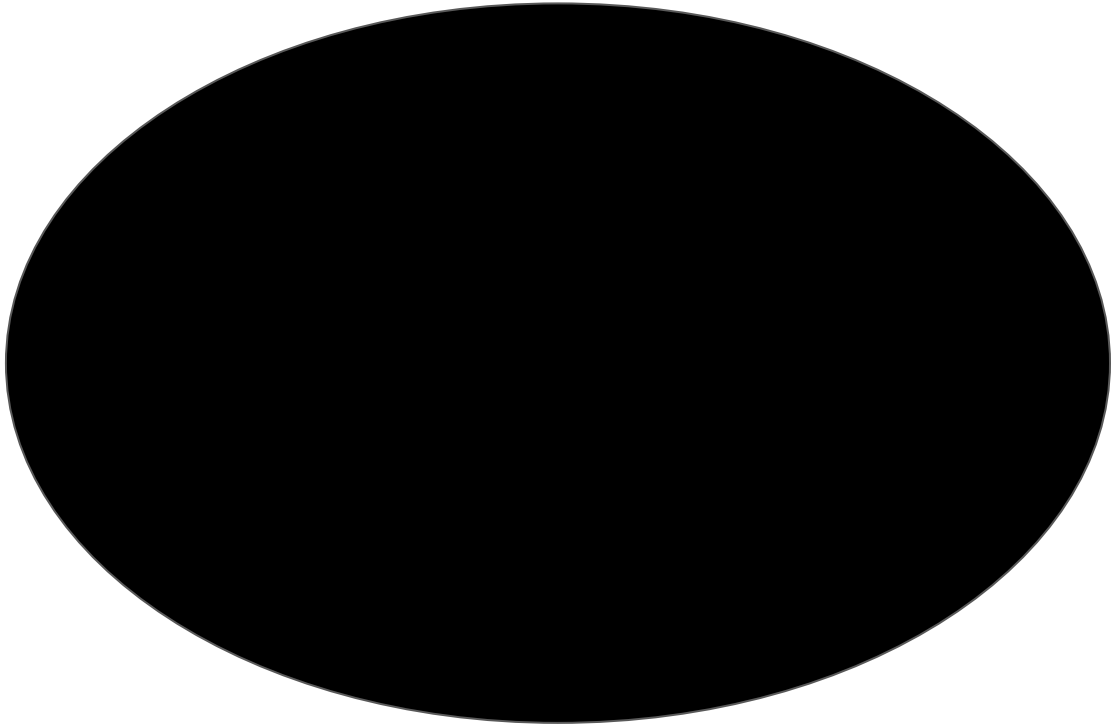
entspricht einem
kleinen Programm mit 10 Klassen,
mit jeweils 10 Methoden



XML - Der Mensch und sein Umgang mit Komplexität II

Durchschnittliche
Komplexitätsgrade
mit denen wir arbeiten.

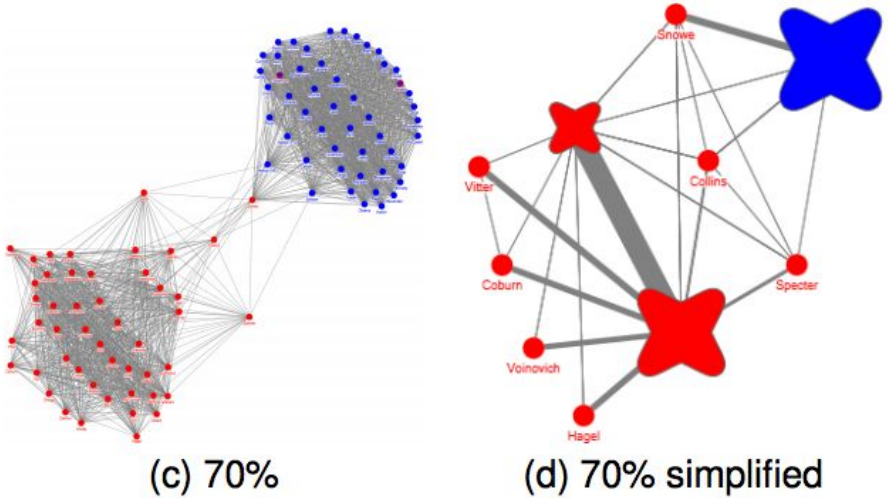
Z.B. Buchhaltungsprozesse



XML - Der Mensch und sein Umgang mit Komplexität III

Wir habe eine Methode entwickelt extreme Komplexitäten zu verstehen indem wir zu folgenden Methoden greifen.

- Klassifizierung
- Aufteilung
- Gruppierung
- Perspektiven
- Umschaltung Perspektive

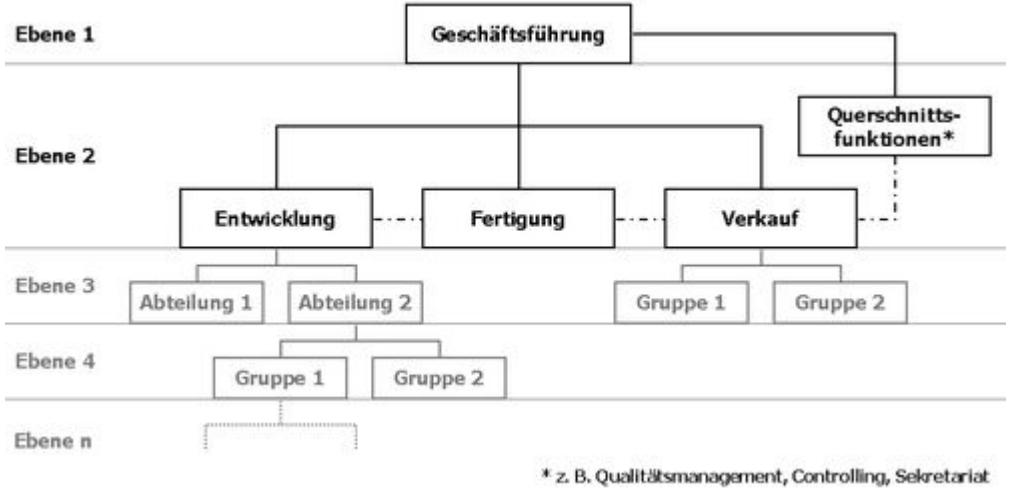


aus medium.com

XML - Der Mensch und sein Umgang mit Komplexität IV

- Bäume sind bei der Komplexitäts-bewältigung eine große Hilfe
aus Wikipedia

XML ist ein Baumformat



XML - Baumbeispiel

1. Person
2. Auto
3. Schulungsraum
4. ...

Datenmodelle
Wie modelliere ich XML?

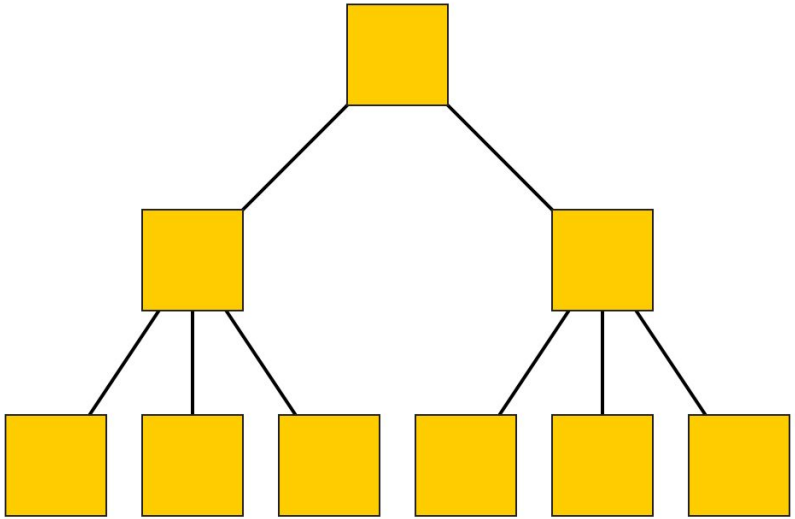
Datenbankmodelle

Abbildung von

- **Statischen Eigenschaften**
  Objekte, Beziehungen und Datentypen
- **Dynamische Eigenschaften**
  Operationen und Beziehungen zwischen Operationen
- **Integritätbedingungen**
  die an Objekten und Operationen hängen

Hierarchische Datenbankmodelle

Modellierung // Hierarchisches Modell (HM)



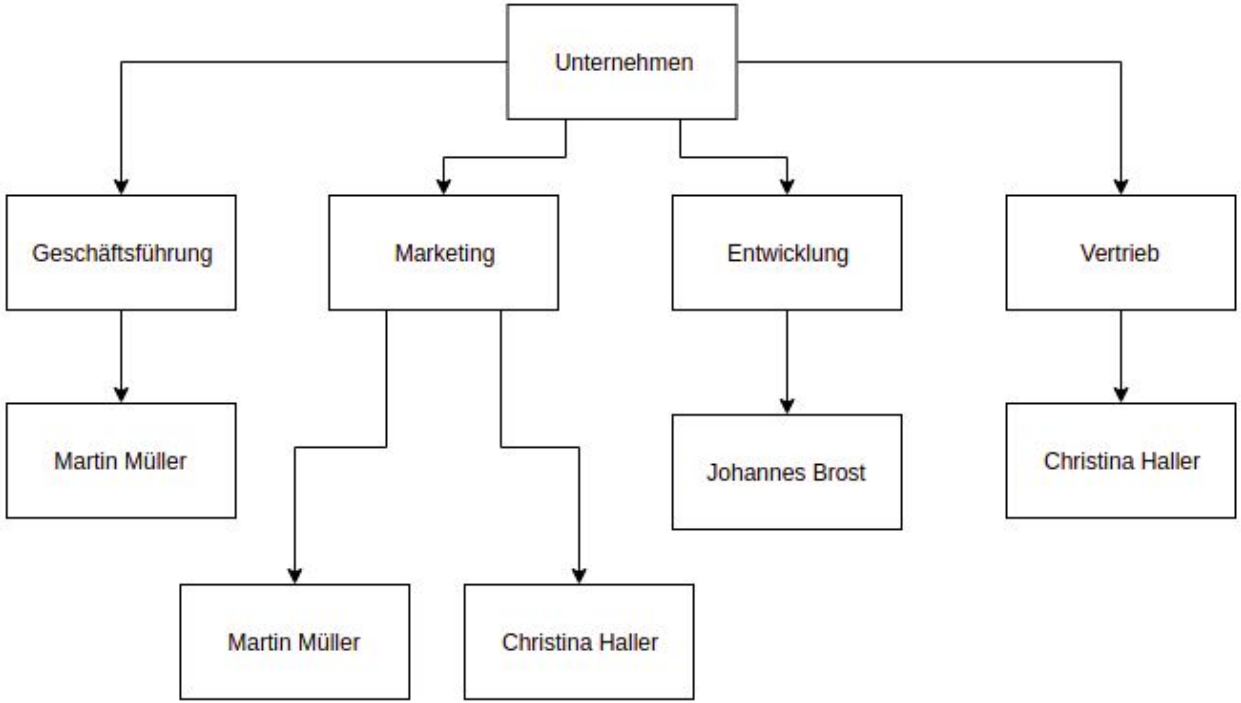
Das **Hierarchische Datenbankmodell** ist das **älteste Datenbankmodell** und wird als hierarchische **Baumstruktur dargestellt**.

aus http://www.datenbanken-verstehen.de

Modellierung // HM // Aufgabe

Erstellen Sie ein HM-Diagram in dem Mitarbeiter Abteilungen zugeordnet werden können.

Modellierung // Hierarchisches Modell (HM) // Beispiel



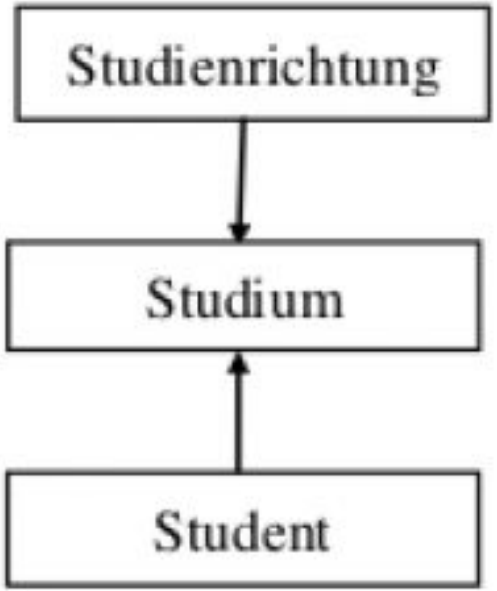
Nur Hierarchieabbildungen mgl. damit ist das Modell sehr starr.

Hohe Redundanz dieses Datenmodells.

Beispiel für hierarchische Datenbanken:
1.Dateisysteme
2. LDAP
3. Content-Repositories
4...

Netzwerk Datenmodelle

Modellierung // Netzwerk Modell (NWM) // I

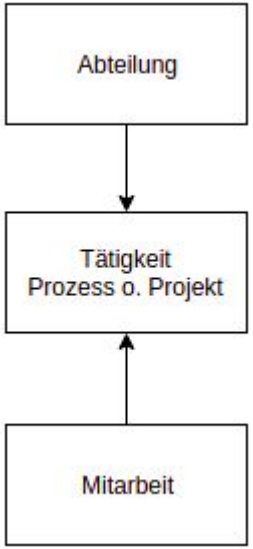


Ein Assoziationsgraph stellt ein Beziehungsknoten mitsamt der beteiligten Entitätet dar.

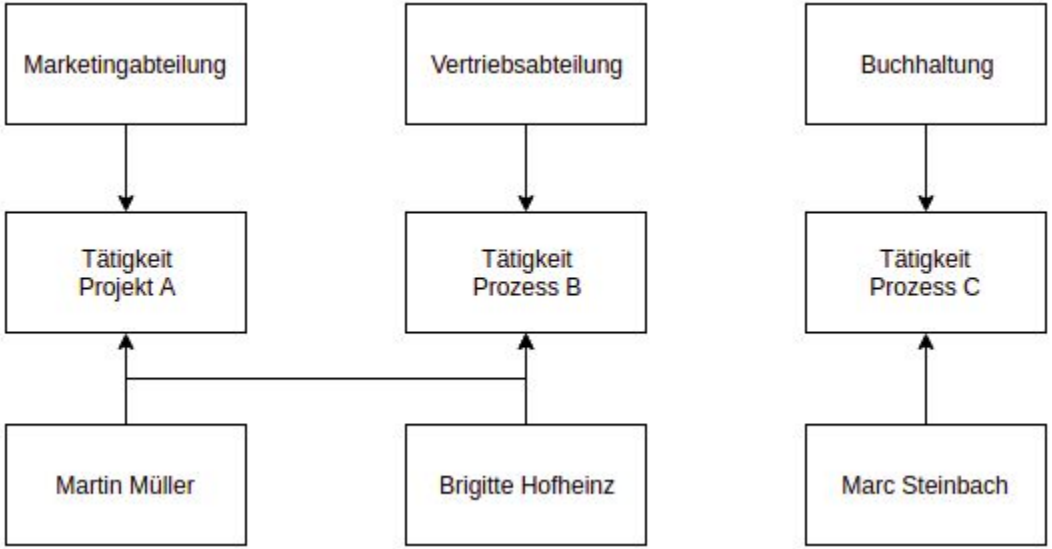
Bei mehreren Assoziationen bietet sich die Erstellung von mehreren Assoziationsgraphen an.

aus proggen.org

Modellierung // Netzwerk Modell (NWM) // Beispiel I // Assoziationsgraph

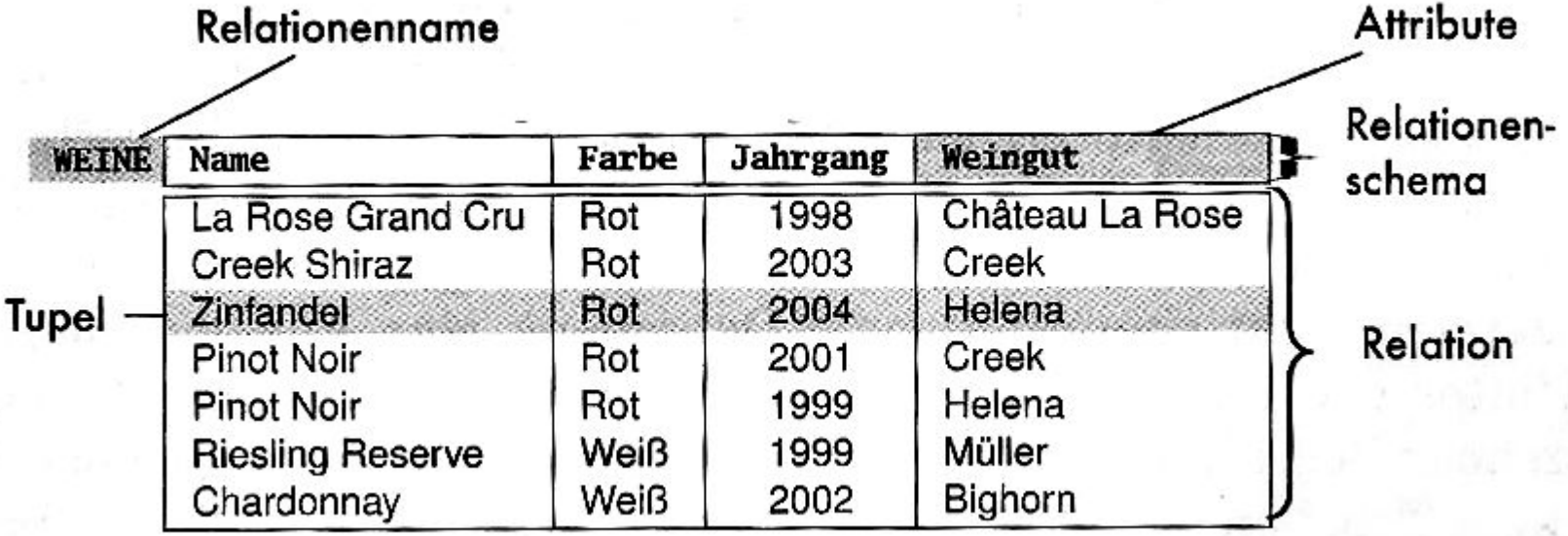


Modellierung // Netzwerk Modell (NWM) // Beispiel II



Entity Relationship Diagramme

Modellierung // Relationenmodell (RM)



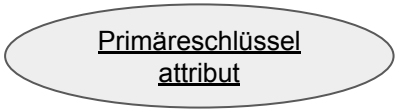
WEINE	Name	Farbe	Jahrgang	Weingut
	La Rose Grand Cru	Rot	1998	Château La Rose
	Creek Shiraz	Rot	2003	Creek
	Zinfandel	Rot	2004	Helena
	Pinot Noir	Rot	2001	Creek
	Pinot Noir	Rot	1999	Helena
	Riesling Reserve	Weiß	1999	Müller
	Chardonnay	Weiß	2002	Bighorn

Entity Relationship Diagramme

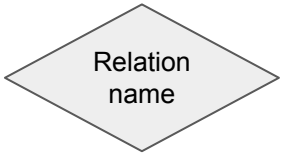
Modellierung // ERD // Elemente

Rechteck als Kennzeichnung eines Tabellenelementes. Der Tabellenname ist im Rechteck enhalten

Ellipse kennzeichnet ein Attribut. Der Attributname ist in der Ellipse enthalten.



Falls der Attributname unterstrichen ist handelt es sich um ein Primärschlüssel.



Stellt eine Relation zwischen Tabellen dar.

Verbindung zwischen Tabelle und Attributen oder Relation

[n..m]

Kardinalitätsbeschreibung

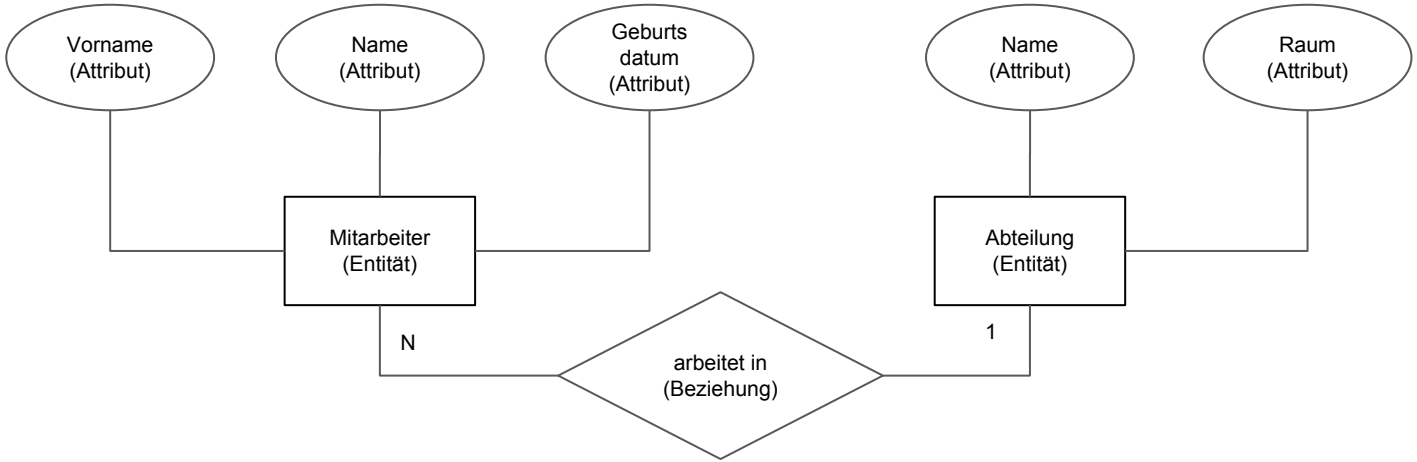
Modellierung // ERD // Aufgabe

Erstellen Sie ein ERD-Diagram in Mitarbeiter Abteilungen zugeordnet werden können.

Folgende Informationen sollen dabei enthalten sein:

1. Mitarbeiter // Name, Vorname, Geburtsdatum
2. Abteilung // Name, Raum

Modellierung // ERD // Elemente



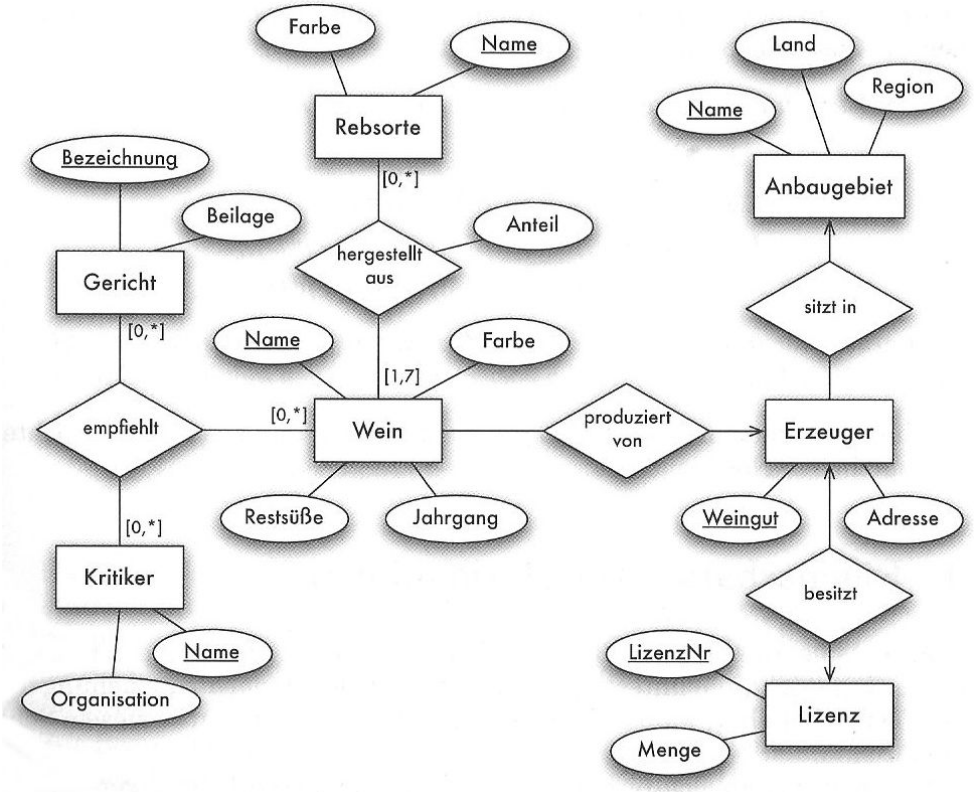
Entität
Eine Informationseinheit. Ein Objekt aus der realen oder der Vorstellungswelt.

Beziehung
Stellt eine Beziehung zwischen ntitäten dar.

Attribut
Definiert eine Eigenenschaft einer Entität dar.

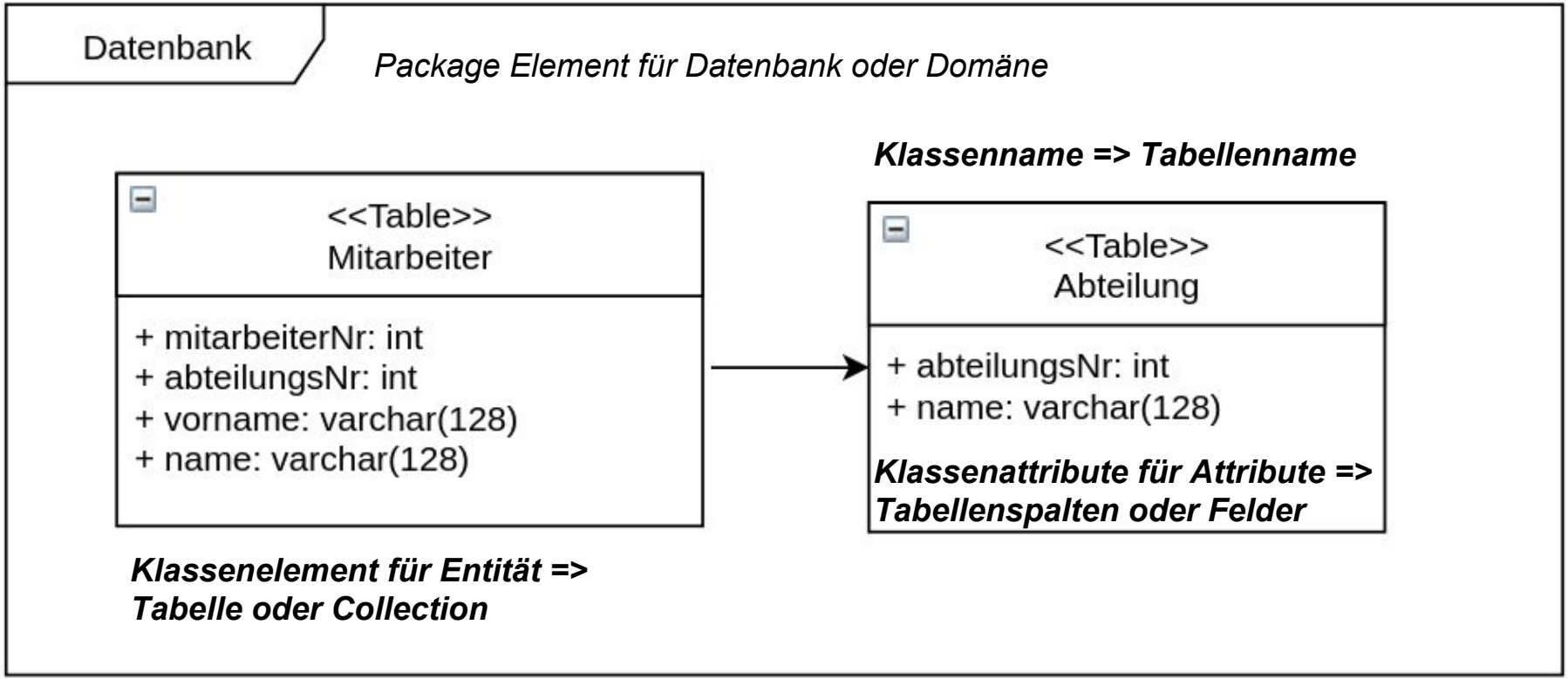
Kardinalität
1:1 Eine Entität zu einer Entität
1:N Eine Entität zu mehreren
N:M Mehrere zu mehreren

Modellierung // Entity Relationship Diagramm (ERD)



UML Diagramme

Modellierung // UML // Elemente



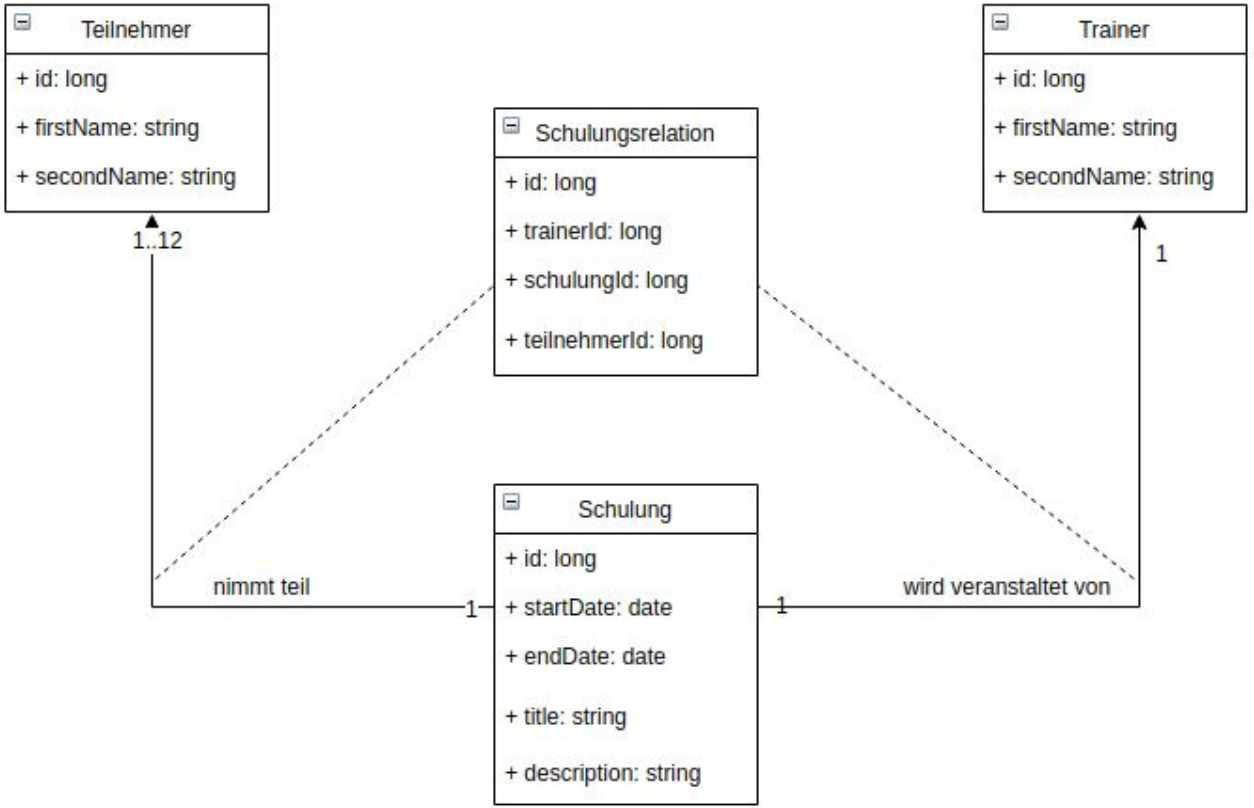
Modellierung // UML // Aufgabe

Erstellen Sie ein UML-Datenbankmodell in dem Schulungen mit Schulungsinhalten, den Teilnehmern sowie den Trainern abgebildet wird.

Folgende Informationen sollen dabei enthalten sein:

1. Trainer // Name, Vorname, Email
2. Schulung // Inhalte, Termine
3. Teilnehmer // Name, Vorname, Email

Modellierung // UML // Aufgabe



XML - Vorheriges Beispiel in UML

1. Person
2. Auto
3. Schulungsraum
4. ...

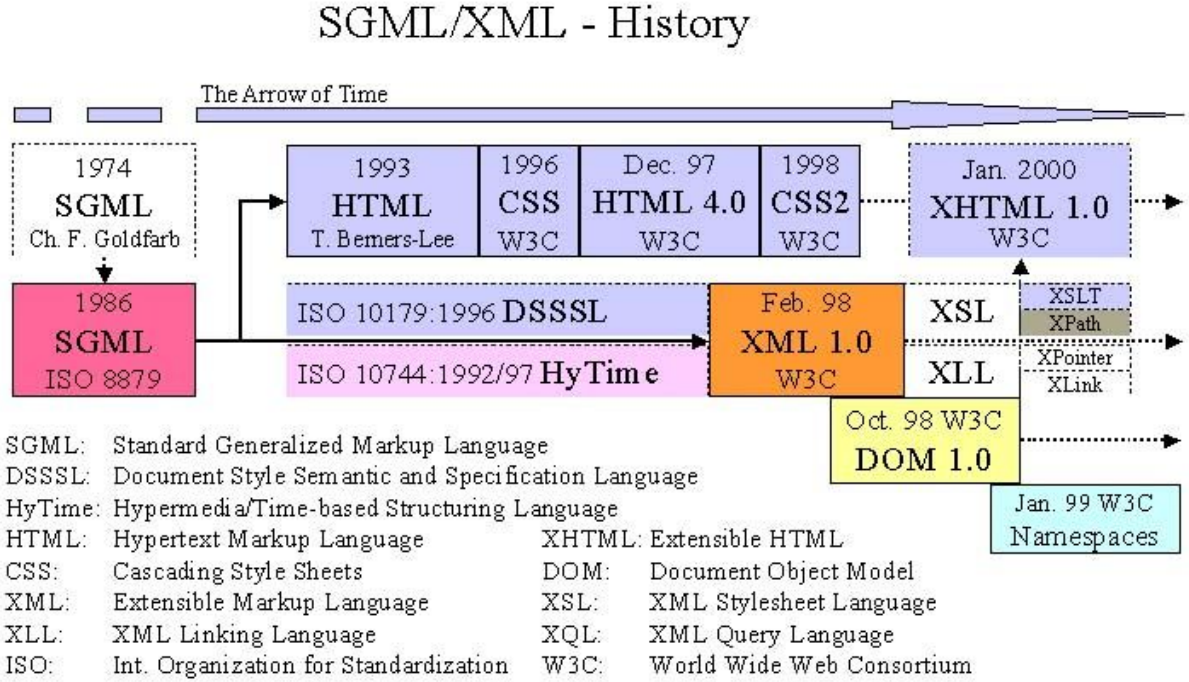
XML-Einführung

XML - Definition

“Die Extensible Markup Language (dt. Erweiterbare Auszeichnungssprache), abgekürzt XML, ist eine Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten im Format einer Textdatei, die sowohl von Menschen als auch von Maschinen lesbar ist.” aus Wikipedia

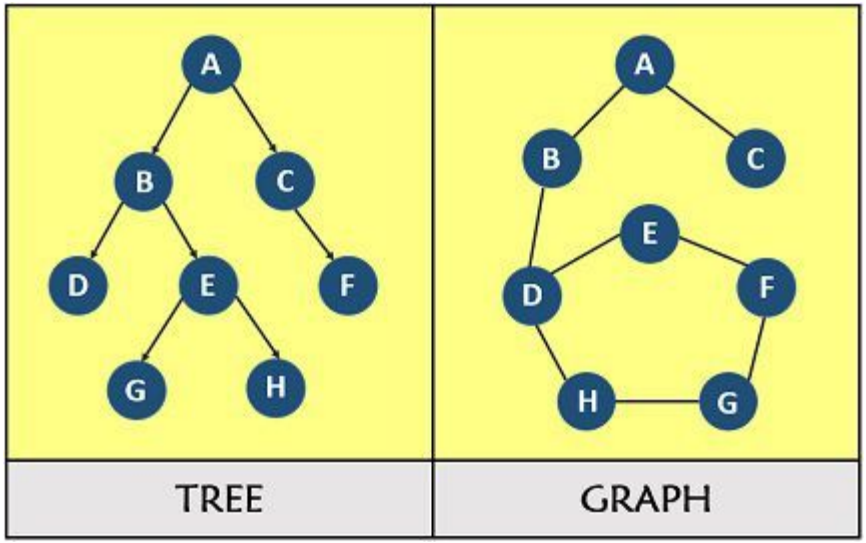
Wichtiges Kennzeichen und der Grund für den Erfolg von XML besteht in seiner Eigenschaft als platformneutrales Format.

XML - Historie



aus jmir.org

XML - Baum vs. Netzwerk



aus techdifferences.com

XML-Grundlagen

XML - Deklaration

Zeigt auf, dass es sich um eine XML-Datei handelt.

Dabei werden Version und Enconding angegeben.

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

sobek innovations

XML & Encoding

	Zeichen		A	ü	π
Zeichensatz	ASCII	Zeichenposition (dezimal)	65	n/a	n/a
	ISO 8859-1		65	252	n/a
	Unicode		65	252	960
Zeichenkodierung	ASCII	Bytewert (hexadezimal)	41	n/a	n/a
	ISO 8859-1		41	FC	n/a
	UTF-8		41	C3 BC	CF 80

aus webkrauts

XML-Elemente - Grundlagen

- ein XML-Dokument enthält immer ein Wurzel-Element
- darf Buchstaben sowie die Sonderzeichen -,_ oder . enthalten
- enthält ein Element Daten besteht er aus einem öffnenden und schließenden Element <element>...</element>
- Elemente ohne Inhalte sehen wie folgt aus <element/>

XML-Elemente - Grundlagen - Beispiel

Beispiel:

```
<info>
    <addresse>123</adresse>
    <kontakt-info>rafael@sobek-innovations.de</kontakt-info>
    <mitarbeiter/>
</info>
```

XML-Elemente - Verschachtelung

XML-Elemente können verschachtelt werden.

Beispiel:

```
<info>
    <addresse>
        <strasse>Kriegsstraße 1</strasse>
        <stadt>Karlsruhe</stadt>
    </adresse>
    <kontakt-info>rafael@sobek-innovations.de</kontakt-info>
    <mitarbeiter/>
</info>
```

XML-Attribute

1. enthalten ihre Inhalte in Anführungszeichen oder Hochkommas
2. befinden sich im Start-Element
3. können auch ohne Inhalt (ohne Anführungszeichen) als Toggle verwendet werden

XML - Wohlgeformtheit

Wohlgeformt ist ein XML-Dokument (englisch well-formed) wenn folgende Regeln eingehalten werden:

1. ein Wurzel-Element
2. ein Start- und ein End-Tag oder ein geschlossener Start-Tag
3. ebenentreu-paarige Verschachtelung
4. Attributinhalte stehen in Anführungszeichen
5. Groß u. Kleinschreibung werden eingehalten

XML-Attribute - Beispiel

Beispiel:

```
<mitarbeiter>
    <person vorname=”Rafael” nachname=”Sobek” entwicklung/>
    <abteilung name=”Personal”></abteilung>
</mitarbeiter>
```

XML - Wohlgeformtheit - Beispiele

```
<e1>
    <e2>
    </e1>
</e2>

<e1>
    <e2>
</e1>

<e1>
    <e2/>
</e1>
```

XML-Datei - Erstellen Sie ein XML-Datei

Aus den Vorherigen Aufgaben

XML vs. JSON

JSON Aufbau

1. Array werden mit “[“ eingeleitet und mit “]” beendet.
2. Objekte werden mit “{” eingeleitet und mit “}” beendet.
3. Objektattribute sehen wie folgt aus:
   a. {“name”: “Rafael”, “vorname”: “Sobek”}
4. Folgende Datentypen sind möglich
   a. null
   b. boolean
   c. string
   d. number

JSON Aufbau

1. Verschachtelung
   a. {
      “name”: “Rafael”
      “info”: {
         “rolle”: “Trainer”
      }
   }

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<notes>
   <note>
      <to>Rafael</to>
      <from>Ferdinand</from>
      <heading>Reminder</heading>
      <body>Denke an die Präsentation!</body>
   </note>
   <note>
      <to>Ferdinand</to>
      <from>Rafael</from>
      <heading>Re: Reminder</heading>
      <body>Ich gebe Sie euch am Mittwoch.</body>
   </note>
</notes>
```

```
[
   {
      "to": "Rafael",
      "from": "Ferdinand",
      "heading": "Reminder",
      "body": "Denke an die Präsentation!"
   },
   {
      "to": "Ferdinand",
      "from": "Rafael",
      "heading": "Re: Reminder",
      "body": "Ich gebe Sie euch am
Mittwoch."
   }
]
```

Aufgabe wandelt bitt folgende XML- in eine JSON-Datei

```
<info>
    <addresse>
        <strasse>Kriegsstraße 1</strasse>
        <stadt>Karlsruhe</stadt>
    </adresse>
    <kontakt-info>rafael@sobek-innovations.de</kontakt-info>
    <mitarbeiter/>
</info>
```

Schematisierung

Schema - Definition

Vergleich Synonym Schablone

“Eine Schablone (von frz. échantillon) ist ein ausgeschnittenes Muster zur Herstellung oder Bearbeitung gleichgestaltiger Dinge.”

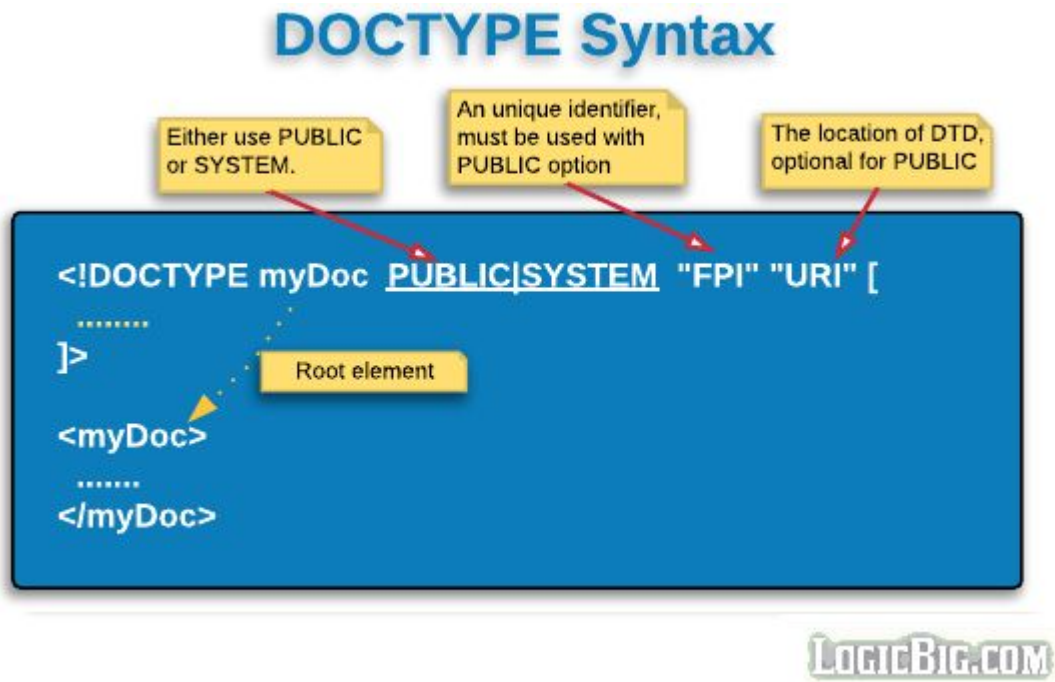
aus Wikipedia

DTD

XML - DTD

“Eine Dokumenttypdefinition (englisch Document Type Definition), abgekürzt DTD, ist ein Satz an Regeln, der benutzt wird, um Dokumente eines bestimmten Typs zu deklarieren.” aus Wikipedia

XML - DTD - Dokumentdeklaration



XML - DTD - Dokumentdeklaration - inline

Innerhalb des XML-Dokumentes nach der XML-Anweisung wird die DTD als Schemadefinition referenziert.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE teilnehmerListe [
]>
<teilnehmerListe>
<!-- schema valider Inhalt -->
</teilnehmerListe>
```

XML - DTD - Elemente

Mit dem DTD-Ausdruck “ELEMENT” wird die Elementenstruktur definiert (damit auch die hierarchische Struktur eines Dokuments von der nicht abgewichen werden darf).

Kardinalitätanweisung:
* beliebig oft, + mindestens einmal, ?keinmal oder genau einmal, keine Angabe genau einmal

Beispiel1 // Liste von teilnehmer-Elementen im Inhalt:

```
<!ELEMENT teilnehmerListe (teilnehmer*)>
```

XML - DTD - Elemente - Beispiel

Bitte unterschiedliche Kardinalitäten ausprobieren.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE teilnehmerListe [
 <!ELEMENT teilnehmerListe (teilnehmer*)>
]>
<teilnehmerListe>
    <teilnehmer></teilnehmer>
</teilnehmerListe>
```

XML - DTD - Elementeverschachtelung - Beispiel

```
<!DOCTYPE teilnehmerListe [
 <!ELEMENT teilnehmerListe (teilnehmer*)>
 <!ELEMENT teilnehmer (person, adresse, kontakt)>
]>
<teilnehmerListe>
    <teilnehmer>
        <person></person>
        <adresse></adresse>
        <kontakt></kontakt>
    </teilnehmer>
</teilnehmerListe>
```

XML - DTD - Attribute

Aufbau:

```
<!ATTLIST element-name
     attribute-name-1 attribute-type-1 attribute-default-value-1
     attribute-name-2 attribute-type-2 attribute-default-value-2
       ...
>
```

DTD Beispiel:

```
<!ATTLIST payment
     type CDATA "check"
     trigger CDATA "middleware"
>
```

XML Beispiel:

```
<payment type="check" trigger=”middleware”  />
```

XML - DTD - Attributetypen

CDATA Beim Wert handelt es sich um eine Zeichenkette (Text)

(en1|en2|..) Liste von möglichen Wert-Inhalten. Bspw. Zahlungsstatus offen oder bezahlt.

ID Beim Wert handelt es sich um ein XML-weiten eindeutigen Wert.

IDREF Der Wert referenziert ein ander Element welches ein ID-Attribut enthält.

IDREFS Der Wert referenziert mehrere andere Elemente die ID-Attribute enthalten.

XML - DTD - Attribute - Beispiel

```
<!DOCTYPE teilnehmerListe [
 <!ELEMENT teilnehmerListe (teilnehmer*)>
 <!ELEMENT teilnehmer (person, adresse, kontakt)>
 <!ATTLIST teilnehmer
     type (mitarbeiter|trainee) "trainee"
     id CDATA #REQUIRED
     >
]>
<teilnehmerListe>
     <teilnehmer id=”1” type=”trainee”>
          <person></person>
          <adresse></adresse>
          <kontakt></kontakt>
     </teilnehmer>
</teilnehmerListe>
```

XML - DTD - Aufgaben

1. Lehrerliste (Lehrer, Klasse, Fach)

2. Mitarbeiterliste (Mitarbeiter, Abteilung, Gehalt)

3. Kundenliste (Kunde, Rechnung, Login)

4. Menükarte (Menü, Kategorie, Inhalte)

5. Bücherregal (Buch, Kategorie, Regal)

XML-Schema

XML - Schema - Definition

“XML Schema, abgekürzt XSD (XML Schema Definition), ist eine Empfehlung des W3C zum Definieren von Strukturen für XML-Dokumente. Anders als bei den klassischen XML-DTDs wird **die Struktur in Form eines XML-Dokuments beschrieben**. Darüber hinaus wird eine **große Anzahl von Datentypen** unterstützt." aus Wikipedia

XML - Schema - Grundlagen

1. Ein XML-Schema wird in sogenannten XSD-Dateien hinterlegt
2. Für ein Element wird das Schema per xsi:schemaLocation referenziert
3. Das Element wird als Schema Instanz (Validierung per XML Schema vorgesehen) definiert

XML - Schema - Verwendung von mehreren Schemas

Mehrere Schemas, und damit Strukturformate, können in eines XML-Datei verwendet werden.

Beispiel:

```
<element
      xmlns="http://example.org/tns"
      xmlns:tns1="http://example.org/tns1"
      xmlns:tns2="http://example.org/tns2"
>
      <element2>element aus ersten Namensraum</element2>
      <tns1:element>anderes element, anderer Namensraum,kein prefix notwendig</tns1:element>
      <tns2:element>anderes element, anderer Namensraum,kein prefix notwendig</tns2:element>
</element>
```

XML - Schema - Skeleton 1 // Bitte erstellen

Einen neuen Ordner (lager) und Datei Lager.xml sowie Lager.xsd anlegen.

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

<tns:waren
     xmlns:tns="http://sobek-innovations.de/Waren"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://sobek-innovations.de/Waren Lager.xsd">

</tns:waren>
```

XML - Schema - Skeleton 2 // Bitte erstellen

Einen neuen Ordner (lager) und Datei Lager.xml sowie Lager.xsd anlegen.

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

<schema xmlns="http://www.w3.org/2001/XMLSchema"

     targetNamespace="http://sobek-innovations.de/Waren"

     xmlns:tns="http://sobek-innovations.de/Waren">

     <element name="waren" type="tns:WarenListe" />

<!-- hier kommen dann die weiteren schema deklarationen -->

</schema>
```

XML - Schema - Einfache Elementendefinition

1. Ein einfaches Element wird wie folgt beschrieben

```
<xs:element name="warenbestand" type="xs:integer"/>
oder
<xs:element name="name" type="xs:string"/>
```

2. Folgende primitive Datentypen stehen zur Auswahl
   xs:string, xs:decimal, xs:integer, xs:boolean, xs:date, xs:time

XML - Schema - Objekt Elementendefinition

Ein komplexes (Verschaltelung von primitiven Elementen zur Strukturierung/Abstraktion von realen Objekten) wird wie folgt beschrieben

```
<xs:element name="Ware">
      <xs:complexType> <!-- leitet ein komplexes struktur ein (Stichwort: Baum) -->
            <xs:sequence> <!-- vorgabe einer Reihenfolge →
                  <!-- primitive Datentypen -->
                  <xs:element name="produktNummer" type="xs:string"/>
                  <xs:element name="regalNummer" type="xs:string"/>
                  <xs:element name="bestand" type="xs:integer"/>
                  <xs:element name="addresse" type="tns:einAnderesKomplexesObject"/>
            </xs:sequence>
      </xs:complexType>
</xs:element>
```

XML - Schema - Attribute

Auch hier können die Standard-Typen verwendet werden => xs:string, xs:decimal, xs:integer, xs:boolean, xs:date, xs:time

```
<xs:complexType name="KontoUeberweisung">

  <xs:attribute name="anweisungsDatum" type="xs:date"/>
  <xs:attribute name="wertstellungsDatum" type="xs:date"/>
  <xs:attribute name="buchungsDatum" type="xs:date"/>

</xs:complexType>
```

XML - Schema - Weitere Einschränkungen (primitive Datentypen) // String

String Beispiel in Lager.xsd:

Wir erstellen einen neuen Datentypen, welcher auf einem primitiven basiert.

```
<xs:simpleType name="Email"></xs:simpleType>
```

XML - Schema - Weitere Einschränkungen (primitive Datentypen) // String II

String Beispiel in Lager.xsd:

Danach weisen wir diesen einen Basistypen zu.

```
<xs:simpleType name="Email">
     <xsd:restriction base="xsd:string">
          <xsd:pattern value="[^@]+@[^\.]+\..+"/>
     </xsd:restriction>
</xs:simpleType>
```

XML - Schema - Weitere Einschränkungen (primitive Datentypen) // Integer I

Integer Beispiel in Lager.xsd:

Der Bestand soll nicht kleiner als 0 und nicht größer als 20 sein.

```
<xs:simpleType name=”Bestand”>
     <xs:restriction base="xs:integer">
          <xs:minInclusive value="0"/>
          <xs:maxInclusive value="20"/>
     </xs:restriction>
</xs:simpleType>
```

XML - Schema - Weitere Einschränkungen (primitive Datentypen) // Integer I

Integer Beispiel in Lager.xsd:

Der Bestand soll nicht kleiner als 0 und nicht größer als 20 sein.

```
<xs:simpleType name=”Bestand”>
     <xs:restriction base="xs:integer">
          <xs:minInclusive value="0"/>
          <xs:maxInclusive value="20"/>
     </xs:restriction>
</xs:simpleType>
```

XML - Schema - Listen

Der Listeninhalt soll nicht kleiner als 0 und nicht größer als 20 sein.

```
<element name="liste" type="tns:EineListe" />

<complexType name="EineListe">
     <sequence>
          <element name="listenElement" type="xs:integer" maxOccurs="unbounded" minOccurs="0"></element>
     </sequence>
</complexType>

Ergebnis:

<liste>
     <listenElement>1</listenElement>
     <listenElement>2</listenElement>
</liste>
```

XML - Schema - Kommentare

Klassisches XML-Kommentar

```
<!-- Das ist ein Kommentar -> wird somit nicht ausgeführt -->
```

XML-Schema-Kommentar

```
<xs:element name="book" type="bookType">
   <xs:annotation>
      <xs:documentation xml:lang="en">A book.</xs:documentation>
   </xs:annotation>
</xs:element>
```

XML - Schema - Beispiel in Lager.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
     targetNamespace="http://sobek-innovations.de/Waren"
     xmlns:tns="http://sobek-innovations.de/Waren">
     <element name="waren" type="tns:WarenListe" />
     <complexType name="WarenListe">
          <sequence>
               <element name="ware" type="tns:WareType"
                    maxOccurs="unbounded" minOccurs="0"></element>
          </sequence>
     </complexType>
     <complexType name="WareType">
          <sequence>
               <element name="produktNummer" type="string" />
               <element name="regalNummer" type="string" />
               <element name="bestand" type="integer" />
          </sequence>
     </complexType>
</schema>
```

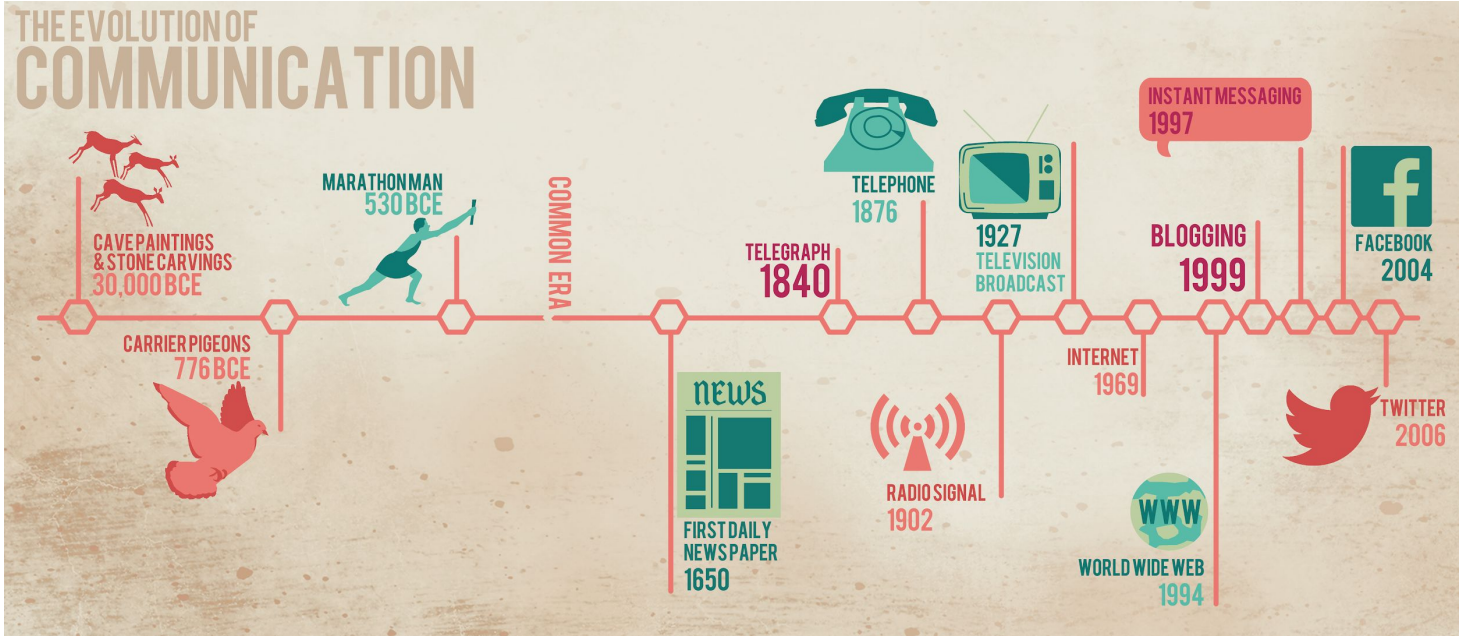
XML - Schema - Aufgaben

1. Einkaufsladen (Adresse, Bereiche(Regale (Produkte)))
2. Haus (Decke, Fundament, Wand (Fenster, Türe))
3. Schule (Klasse, Zimmer, Schüler (Noten))
4. Computer (Gehaeuse, Hauptplatine (CPU, RAM), Festplatte)
5. Bibliothek (Bereiche (Regale(Bücher)))

HTML

Das bekannteste XML-Format

Einführung // Die Evolution der Kommunikation



Quelle:
https://sci10sectionm.wordpress.com/2013/12/08/the-evolution-of-communication-effects-on-the-world-of-science/

Einführung // Das Internet - Start 1973/74

<table>
<tr><th colspan="2">OSI Layer</th><th>TCP/IP Layer</th><th colspan="2">TCP/IP Protocols</th></tr>
<tr><td>7</td><td>Application</td><td rowspan="3">Application Layer</td><td rowspan="3">Telnet FTP SMTP DNS SNMP</td><td>NFS</td></tr>
<tr><td>6</td><td>Presentation</td><td>XDR</td></tr>
<tr><td>5</td><td>Session</td><td>RPC</td></tr>
<tr><td>4</td><td>Transport</td><td>Transport Layer</td><td>TCP</td><td>UDP</td></tr>
<tr><td>3</td><td>Network</td><td>Internet Layer</td><td colspan="2">RIP, OSPF, EGP
IP, ICMP, ARP, RARP</td></tr>
<tr><td>2</td><td>Data Link</td><td rowspan="2">Network Interface Layer</td><td colspan="2" rowspan="2">Ethernet, FDDI, Frame Relay, ATM, SLIP, PPP</td></tr>
<tr><td>1</td><td>Physical</td></tr>
</table>

Quelle: https://www.itwissen.info/IP-Internet-protocol-IP-Protokoll.html

Einführung // WWW - World Wide Web

HTML: HyperText Markup Language. The markup (formatting) language for the web.

URI: Uniform Resource Identifier. A kind of “address” that is unique and used to identify to each resource on the web. It is also commonly called a URL.

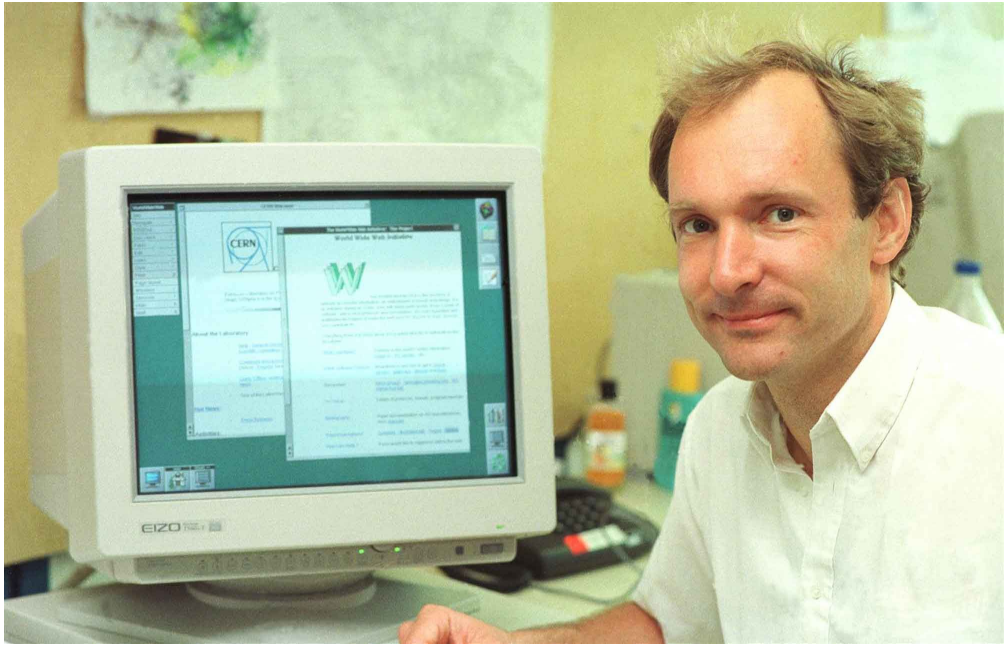
HTTP: Hypertext Transfer Protocol. Allows for the retrieval of linked resources from across the web.

Quelle: https://webfoundation.org/about/vision/history-of-the-web/

Einführung // WWW - Abgrenzung Internet

- IRC, E-Mail, Telnet etc.
- Übertragung und Verknüpfung von elektronischen Dokumenten
- Internet != WWW
- WWW basiert auf der Internettechnologie (TCP und IP)

Einführung // Tim Berners-Lee



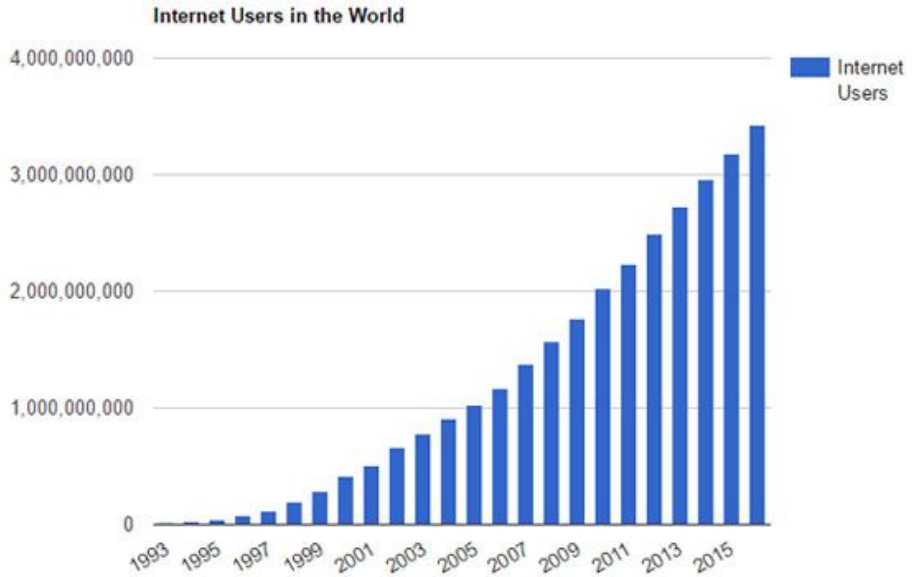
1989 March
“Information Management: A Proposal”

1990 Oct
Entwicklung von HTML, URI und HTTP

1990 Dez
Entwicklung des ersten Webbrowsers u. -Servers

Quelle: https://webfoundation.org/about/vision/history-of-the-web/

Einführung // Die nachfolgende Expansion

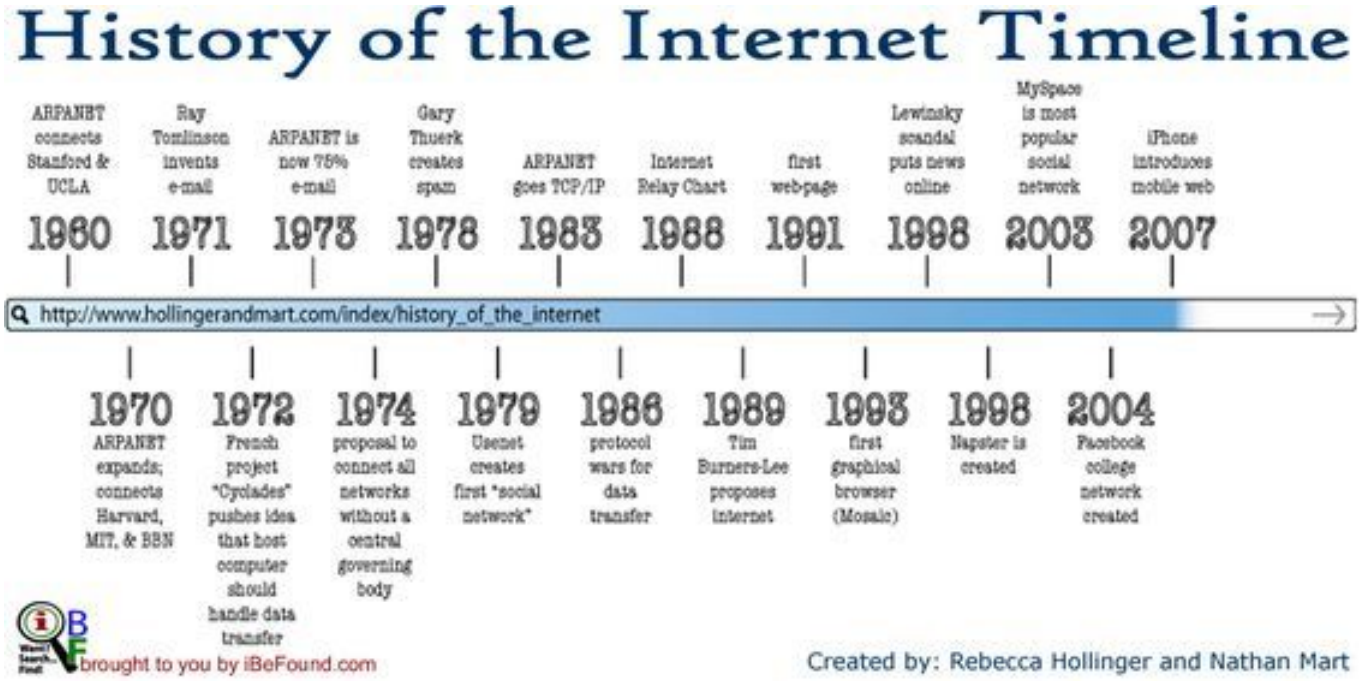


Quelle: http://www.codeholic.in/happy-world-wide-web-day/

Einführung // HTML - Was ist das?

- HTML ist ein Bestandteil des World Wide Webs (WWW)
- Dient der Strukturierung von Texten und Bildern
- Enthält Metainformationen
- Dient der semantische Anreicherung von Inhalten
- Abgrenzung zu CSS: HTML ist nicht verantwortlich für die Formatierung

Einführung // Historie Übersicht

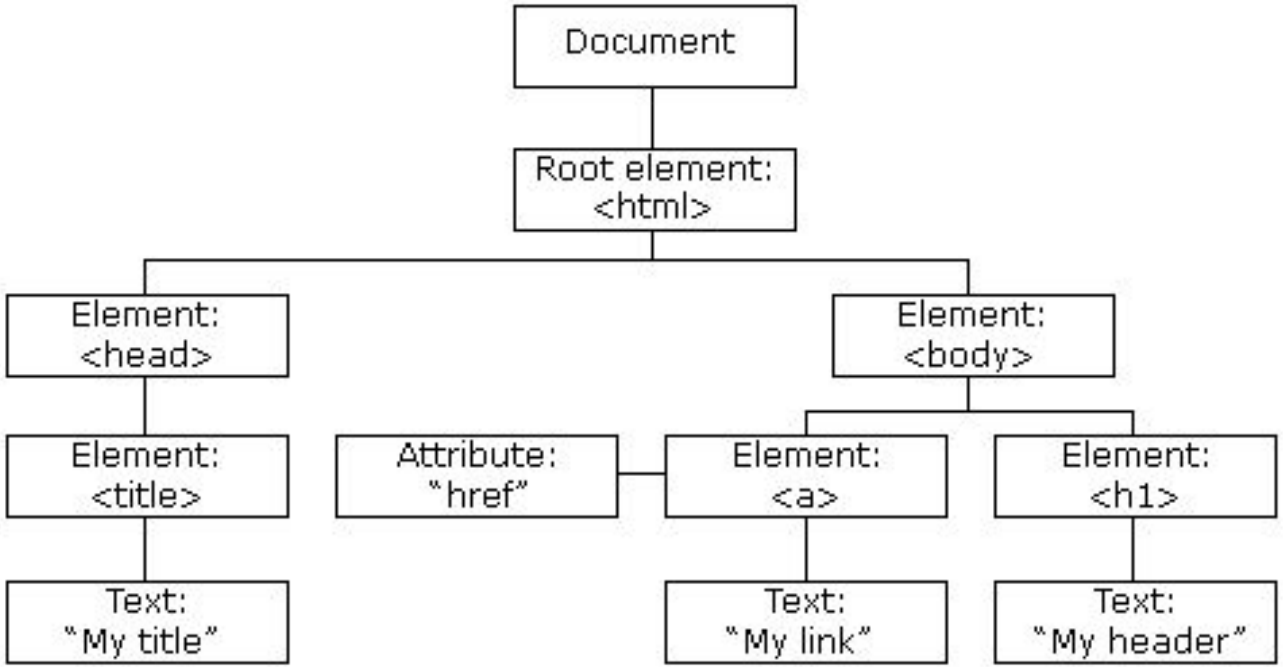


Quelle: https://www.pinterest.de/pin/221591244142836968

Hypertext Markup Language (HTML)

- Textbasierte Auszeichnungssprache
- Dient der Strukturierung von Texten und Bildern
- Enthält Meta-Informationen
- Dient der semantische Anreicherung von Inhalten
- Hyperlinks dienen der Verkünpfung von Dokumenten
- Abgrenzung zu CSS: HTML ist nicht verantwortlich für die Formattierung

DOM // Document Object Model



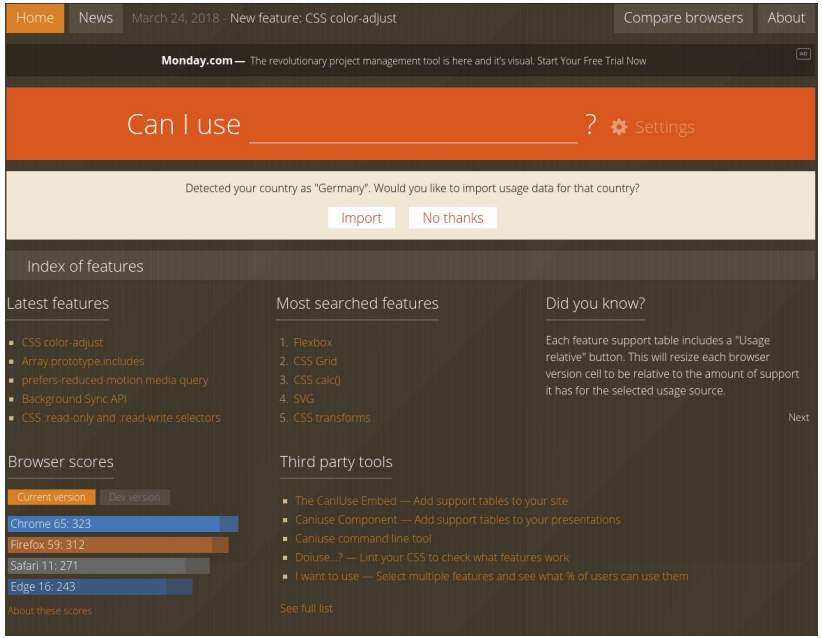
Bestandteile

- Elemente
- Attribute
- Textknoten

Quelle: https://www.w3schools.com/js/js_htmldom.asp

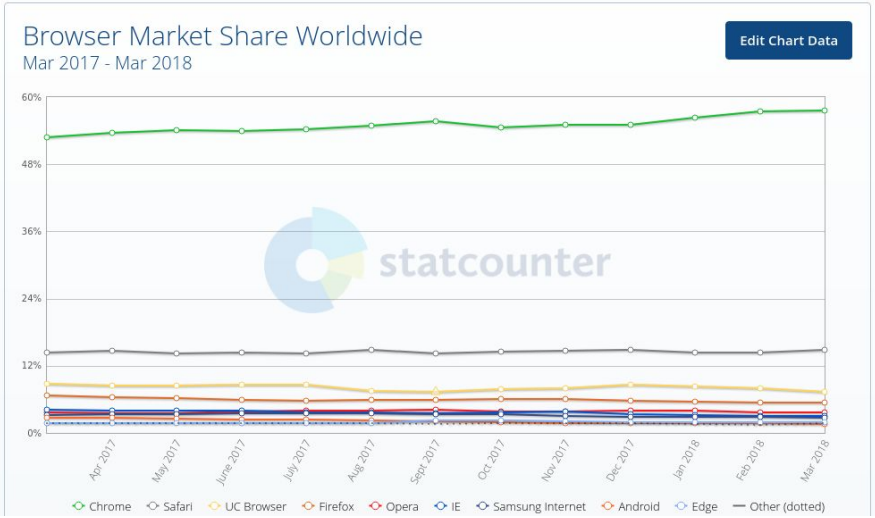
Browserkompatibilität // caniuse.com

Es gibt eine Vielzahl von CSS-Optionen, die nicht alle von jeden Browser unterstützt werden. caniuse.com unterstützt euch dabei zu überprüfen, wie weit eure CSS-Regeln unterstützt werden.

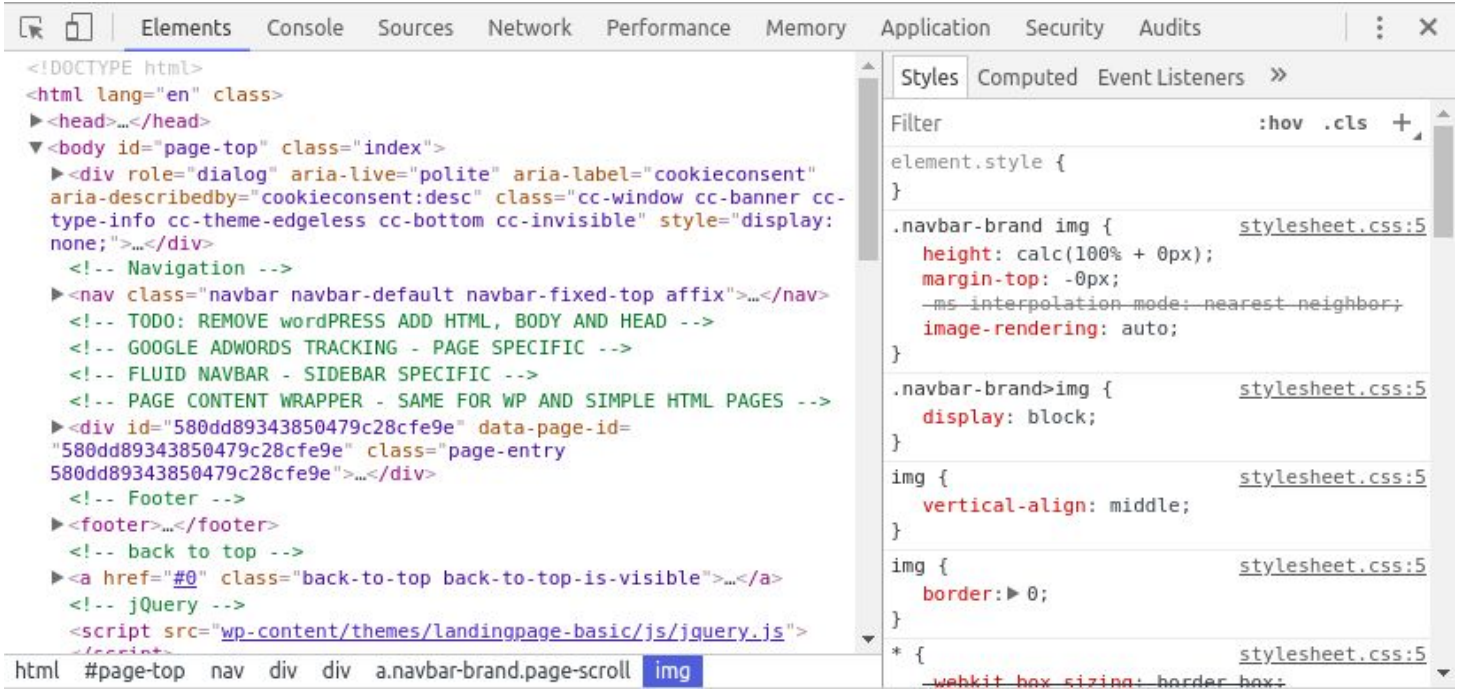


Browserkompatibilität // gs.statcounter.com

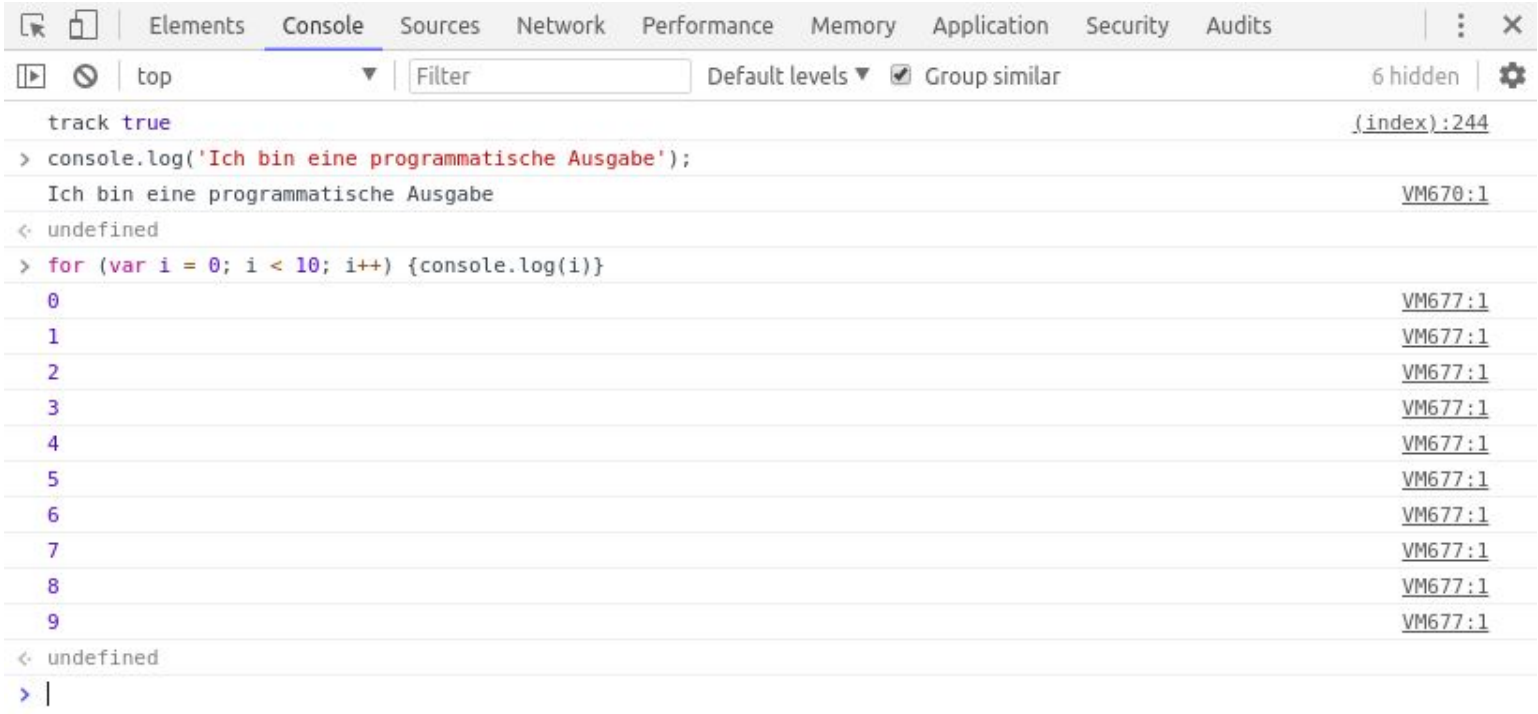
Welche Browser und in welcher Version gerade bei den Nutzer Verwendung finden könnt ihr gs.statcounter.com rausfinden.



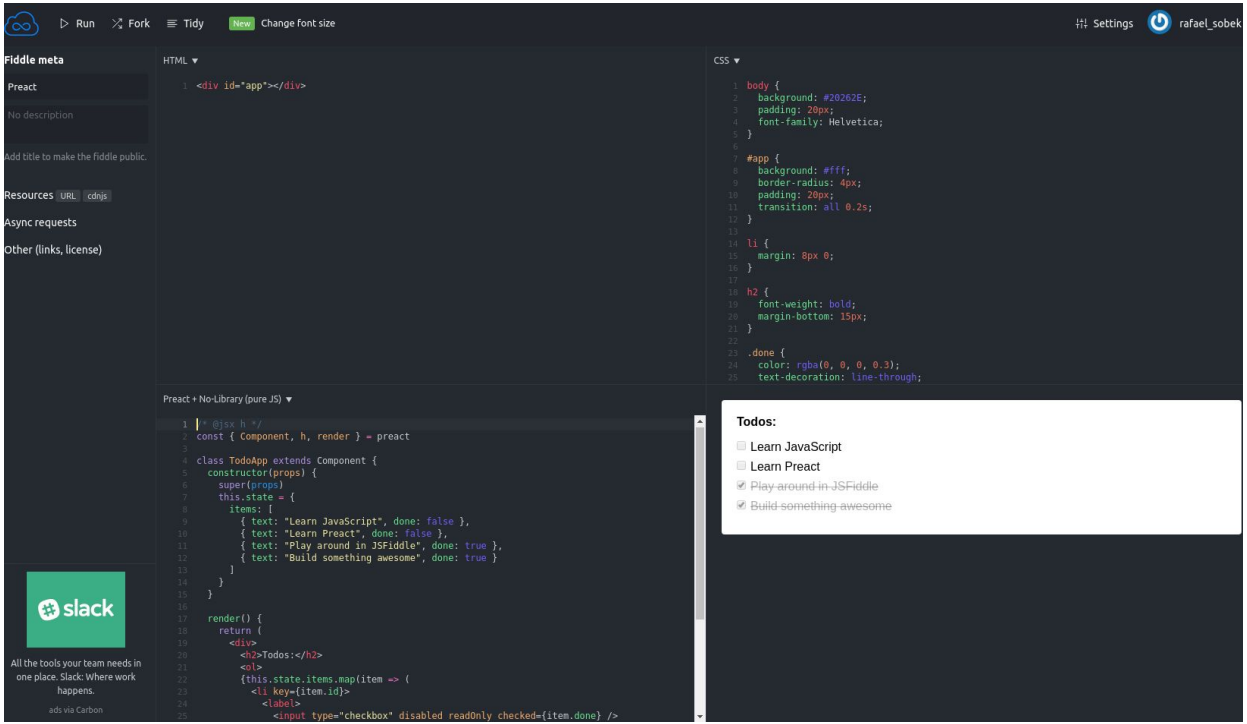
Webconsole Chrome - DOM & CSS Explorer



Webconsole Chrome - Console



Tools // jsfiddle.net



Syntax // Aufbau Elemente

Elemente - **Tags**

Beispiel: **<element></element> oder <element/>**

Elemente - **Attribute**

Beispiel: **<element attr1=”123”></element> oder <element attr2=”321”/>**

Elemente - **Textknoten**

Beispiel: <element>**test1knoten**</element>**test2knoten**

Syntax // Adressierung von Elementen

Elemente - **ID-Adressierung**

Beispiel: **<element id=”123”></element>**

Bemerkung: Wert kann nur **einmal** vergeben werden (Stichwort: Eindeutigkeit).

Elemente - **Class-Adressierung**

Beispiel: **<element class=”123”></element> und <element class=”123”/>**

Bemerkung: Wert kann nur **mehrfach** vergeben werden.

Syntax // Wichtiges - Übersicht

DOCTYPE - Deklaration

Ein HTML Dokument wird immer mit der DOCTYPE Deklaration eingeleitet

Beispiel: **<!DOCTYPE html>**

Elemente ohne Kindelemente

Es gibt Elemente die keine Kindelemente enthalten. Sie dürfen aber ebenfalls Attribute enthalten.

Beispiel: **
**

Syntax // Wichtiges - Übersicht

Attribute

Attribute werden mit Anführungszeichen eingeleitet und abgeschlossen.

Beispiel: **W3C**

Beispiel 1: Einfache HTML-Datei er

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>HTML Beispiel</title>
  </head>
  <body>
    Lorem ipsum dolor sit amet, consetetur sadipscing elitr,
sed diam nonumy eirmod tempor invidunt ut labore et dolore
magna aliquyam erat, sed diam voluptua.
  </body>
</html>
```

Auszeichnung // Wurzelelement

Das Wurzelelement

<html> Steht für den Wurzelknoten eines HTML- oder XHTML-Dokuments. Alle weiteren Elemente müssen Nachkommen dieses Elements sein.

Quelle: https://developer.mozilla.org/de/docs/Web/HTML/HTML5/HTML5_element_list

Auszeichnung // Head-Elemente

Unter anderem werden hier die Metadaten des Dokuments definiert.

<head> Bezeichnet eine Sammlung von Metadaten des Dokuments. Hierzu gehören auch Links zu oder Definitionen von Skripts und Stylesheets.

<title> Definiert den Titel eines Dokuments, der in der Titelzeile des Browsers im Tab der betreffenden Seite angezeigt wird. Darf ausschließlich Text enthalten. Eventuell enthaltene Tags werden nicht interpretiert.

<meta> *Wird für die Definition von Metadaten verwenden, die mit keinem anderen HTML-Element definiert werden können.*

Quelle: https://developer.mozilla.org/de/docs/Web/HTML/HTML5/HTML5_element_list

Auszeichnung // Head-Elemente

Unter anderem werden hier die Metadaten des Dokuments definiert.

<link> Wird verwendet, um externe JavaScript- und CSS-Dateien in das aktuelle HTML-Dokument einzubinden.

<meta> Wird für die Definition von Metadaten verwenden, die mit keinem anderen HTML-Element definiert werden können.

<style> Tag für die Definition eines internen CSS-Stylesheets.

Quelle: https://developer.mozilla.org/de/docs/Web/HTML/HTML5/HTML5_element_list

Auszeichnung // Scripting-Elemente

<script> Definiert entweder ein internes Skript oder einen Link auf ein externes Skript. Als Programmiersprache wird JavaScript verwendet.

<noscript> Definiert alternative Inhalte, die angezeigt werden sollen, wenn der Browser kein Skripting unterstützt.

Quelle: https://developer.mozilla.org/de/docs/Web/HTML/HTML5/HTML5_element_list

Auszeichnung // Body-Elemente

<body> HTML-Inhalt
Steht für den Hauptinhalt eines HTML-Dokuments. Jedes Dokument kann nur ein <body>-Element enthalten.

<section> Abschnitt (HTML5)
Beschreibt einen Abschnitt eines Dokuments.

<nav> Navigation (HTML5)
Beschreibt einen Abschnitt der ausschließlich Navigationslinks enthält.

Quelle: https://developer.mozilla.org/de/docs/Web/HTML/HTML5/HTML5_element_list

Auszeichnung // Body-Elemente

<article> Artikel (HTML5)
Beschreibt eigenständigen Inhalt, der unabhängig von den übrigen Inhalten sein kann.

<aside> Bemerkung (HTML5)
Steht für eine Randbemerkung. Der übrige Inhalt sollte auch verständlich sein, wenn dieses Element entfernt wird.

<h1>,<h2>,<h3>,<h4>,<h5>,<h6> Titel (häufiger Einsatz)
Hiermit werden Überschriften definiert.

Quelle: https://developer.mozilla.org/de/docs/Web/HTML/HTML5/HTML5_element_list

Auszeichnung // Body-Elemente

<header> Kopfzeile (HTML5)
Definiert den Kopfteil ("header") einer Seite oder eines Abschnitts. Er enthält oft ein Logo, den Titel der Website und die Seitennavigation.

<footer> Fußzeile (HTML5)
Definiert den Fußteil ("footer") einer Seite oder eines Abschnitts. Er enthält oft Copyright-Hinweise, einen Link auf das Impressum oder Kontaktadressen.

<address> Adresse
Definiert einen Abschnitt mit Kontaktinformationen.

Quelle: https://developer.mozilla.org/de/docs/Web/HTML/HTML5/HTML5_element_list

Auszeichnung // Body-Elemente

<main> Hauptinhalt (HTML5)
Definiert den Hauptinhalt der Seite. Es ist nur ein <main> Element pro Seite zulässig.

Quelle: https://developer.mozilla.org/de/docs/Web/HTML/HTML5/HTML5_element_list

Auszeichnung // Gruppierungselemente

<p> Ein Absatz (häufiger Einsatz)
Der Inhalt dieses Elements soll als Absatz dargestellt werden.

<hr> Eine Trennlinie
Bezeichnet einen thematischen Bruch zwischen Absätzen eines Abschnitts, Artikels oder anderem längeren Inhalt.

<pre> Vorformatierter Inhalt
Zeigt an, das der Inhalt dieses Elements vorformatiert ist und das dieses Format erhalten bleiben soll.

<blockquote> Zitat
Kennzeichnet ein Zitat.

Quelle:
https://developer.mozilla.org/de/docs/Web/HTML/HTML5/HTML5_element_list

Auszeichnung // Gruppierungselemente

<blockquote> Zitat
Kennzeichnet ein Zitat.

** Geordnete Liste**
Definiert eine geordnete Liste, bei der die Einträge eine bestimmte Reihenfolge haben müssen.

** Ungeordnete Liste**
Definiert eine Liste ungeordneter Einträge.

Quelle: https://developer.mozilla.org/de/docs/Web/HTML/HTML5/HTML5_element_list

Auszeichnung // Gruppierungselemente

** Listeneintrag**
Kennzeichnet einen Listeneintrag. Diesem wird oftmals ein Aufzählungszeichen ("bullet") vorangestellt.

<dl> Definitionsliste
Kennzeichnet eine Definitionsliste aus Begriffen und den dazugehörigen Definitionen.

<dt> Kennzeichnung f. Begriffsdefinition
Kennzeichnet einen Begriff der im folgenden <dd>-Element beschrieben wird.

Quelle: https://developer.mozilla.org/de/docs/Web/HTML/HTML5/HTML5_element_list

Auszeichnung // Gruppierungselemente

<dd> Definitionsbeschreibung
Markiert die Definition des oder der Begriffe, die in den direkt vorangehenden <dt>-Element angegeben wurden.

<figure> Kennzeichnung Abbildung (HTML5)
Kennzeichnet eine Abbildung, die einen Teil des Dokuments illustriert.

<figcaption> Abbildungbeschriftung (HTML5)
Bezeichnet die Beschriftung einer Abbildung.

Quelle: https://developer.mozilla.org/de/docs/Web/HTML/HTML5/HTML5_element_list

Auszeichnung // Gruppierungselemente

** Allgemeiner Container (häufiger Einsatz)**

Bezeichnet ein allgemeines Container-Element ohne spezielle semantische Bedeutung. Wird oft zusammen mit class- oder id-Attributen verwendet, um es in Skripts oder Stylesheets auswählen zu können.

Quelle: https://developer.mozilla.org/de/docs/Web/HTML/HTML5/HTML5_element_list

Auszeichnung // Semantik Text

<a> Link (häufiger Einsatz)
Bezeichnet einen Hyperlink , der auf eine andere Ressource verweist (angegeben durch das href-Attribut).

** & Hevorhebung**
Steht für hervorgehobenen Text.

<small> Das Kleingedruckte
Steht für das "Kleingedruckte" eines Dokuments, wie Ausschlussklauseln, Copyright-Hinweise oder andere Dinge, die für das Verständnis des Dokuments nicht unbedingt nötig sind.

Quelle: https://developer.mozilla.org/de/docs/Web/HTML/HTML5/HTML5_element_list

Auszeichnung // Semantik Text

<abbr> Abkürzung
Bezeichnet eine Abkürzung oder ein Akronym .

<sub>,<sup> Tief u. -hochgestellter Text
Markiert tiefgestellten , bzw. hochgestellten Text .

<i>, <b>, <u> Abgrenzung zum übrigen Text
Kursiv, Fett oder Unterstrichen

<mark> Referenz (HTML5)
Abgrenzung auf Grund von Referenzgründen.
Quelle: https://developer.mozilla.org/de/docs/Web/HTML/HTML5/HTML5_element_list

Auszeichnung // Eingebettete Elemente

<img> Bild (häufiger Einsatz)

<iframe> Einbettung (häufiger Einsatz)
Einbettung fremde Seite

<video> Videodatei (HTML5 & häufiger Einsatz)
Steht für eine Videodatei und die dazugehörigen Audiodateien, sowie die für das Abspielen nötigen Kontrollelemente.

<audio> Audiodatei (HTML5)
Markiert eine Tondatei oder einen Audiostream.

Quelle: https://developer.mozilla.org/de/docs/Web/HTML/HTML5/HTML5_element_list

Auszeichnung // Eingebettete Elemente

<img> Bild (häufiger Einsatz)

<iframe> Einbettung (häufiger Einsatz)
Einbettung fremde Seite

<video> Videodatei (HTML5 & häufiger Einsatz)
Steht für eine Videodatei und die dazugehörigen Audiodateien, sowie die für das Abspielen nötigen Kontrollelemente.

<audio> Audiodatei (HTML5)
Markiert eine Tondatei oder einen Audiostream.

Quelle: https://developer.mozilla.org/de/docs/Web/HTML/HTML5/HTML5_element_list

Auszeichnung // Eingebettete Elemente

<canvas> Bitmap (häufiger Einsatz)
Steht für einen Bitmap-Bereich, der von Skripts verwendet werden kann, um beispielsweise Diagramme, Spielegraphiken oder andere visuellen Effekte dynamisch darzustellen.

<svg> Vektorgrafik (häufiger Einsatz)
Definiert eine eingebettete Vektorgrafik.

Quelle: https://developer.mozilla.org/de/docs/Web/HTML/HTML5/HTML5_element_list

Auszeichnung // Tabellen

<table> Tabelle (häufiger Einsatz)
Markiert eine Tabelle, d.h. Daten mit mehr als einer Dimension.

<tr> Tabellenzeile (häufiger Einsatz)
Markiert die Gruppe der Tabellenzeilen, die die Beschriftungen der Tabellenspalten enthalten.

<th> o. <td> Spaltenüberschrift oder Tabellenzelle (häufiger Einsatz)
Kennzeichnet Spaltenbeschriftung o. eine einzelne Tabellenzelle.

Quelle: https://developer.mozilla.org/de/docs/Web/HTML/HTML5/HTML5_element_list

Auszeichnung // Formulare

<form> Formular Wurzelelement (häufiger Einsatz)
Formulare bestehen typischerweise aus einer Reihe von Kontrollelementen, deren Werte zur weiteren Verarbeitung an einen Server übertragen werden.

<input> Eingabefeld (häufiger Einsatz)
Steht für ein Feld für Benutzereingaben eines bestimmten Typs. Der Typ (Radiobutton, Ankreuzfeld, Texteingabe, etc.) wird anhand des type-Attributs angegeben.

<select> Auswahlfeld (häufiger Einsatz)
Kennzeichnet ein Kontrollelement mit einer von Optionen.

Quelle: https://developer.mozilla.org/de/docs/Web/HTML/HTML5/HTML5_element_list

Auszeichnung // Formulare

<textarea> Eingabefeld mehrzeilig (häufiger Einsatz)

Markiert ein Element für mehrzeilige Texteingaben .

Quelle: https://developer.mozilla.org/de/docs/Web/HTML/HTML5/HTML5_element_list

XPath

Suche in XML-Dokumenten

XPath - Definition

“Die XML Path Language (XPath) ist eine vom W3-Konsortium entwickelte Abfragesprache, um Teile eines XML-Dokumentes zu adressieren und auszuwerten.”

aus Wikipedia

XPath - Beispiel bitte erstellen

1. neuen Ordner xpath erstellen
2. neue XML-Datei erstellen

```
<?xml version="1.0" encoding="UTF-8"?>
<buecherladen>
  <buch>
    <titel lang="en">Glashaus</titel>
    <autor>Charles Stross</autor>
    <preis>29.99</preis>
  </buch>
  <buch>
    <titel lang="de">Das neue Buch Genesis</titel>
    <autor>Bernard Beckett</autor>
    <preis>10.00</preis>
  </buch>
</buecherladen>
```

XPath - Anweisungen I

knotenname	Selektiert alle Elemente mit diesen Knotennamen Bspw.: book
/	Selektiert vom Wurzelelement aus drunterliegende Elemente
//	Ausgehend vom aktuellen Knoten alle die in den drunterligenden Hierachien sich befinden
.	Selektiert den aktuellen Knoten
..	Selektiert den drüberliegenden Knoten zum Aktuellen
@	Selektiert ein Attribut

XPath - Anweisungen II

/parent/child[1]	Elemente per Index identifizieren
/parent/child[last()]	Das letzte Element referenzieren
/parent/child[position()<5]	Nur Elemente bis 5 referenzieren
/parent/child[@attributName='true']	Nur Kindelemente referenzieren in dem ein Attribut mit true gesetzt ist
/parent/child[value>15]/subelement	Nur Unterelemente von drüberliegenden ausgeben, wenn diese Werte über 15 aufweisen
/parent/*	Alle Elemente unter parent auswählen
/parent/child[contains(., 'test')]	Alle Childelemente die test enthalten
/parent/child[contains(., 'test') or contains(., 'wasanderesw')]	Auch komplexere Bedingungen sind möglich.
count(//child)	Die Anzahl aller Childs aller Parents

XPath Aufgabe

1. Newsfeed Datei von heise runterladen

   https://www.heise.de/newsticker/heise-atom.xml

2. Aufgaben
   a. Alle Titel ausgeben in denen Linux enthalten ist
      -> Vorstellung der Ergebnisse
   b. Die Anzahl der Titel ausgeben, die heute herausgegeben worden sind?
      -> Vorstellung der Ergebnisse
   c. Alle Artikel-Links ausgeben für Artikel bei denen im
      “content”-Element Linux enthalten ist.
      -> Vorstellung der Ergebnisse
   d. Alle Artikel in denen das publishDate und updateDate gleich sind
      -> Vorstellung der Ergebnisse

XSLT

XML in andere XML o. TXT-Formate wandeln

XSLT - Definition

“XSL Transformation, kurz XSLT, ist eine Programmiersprache zur Transformation von XML-Dokumenten. Sie ist Teil der Extensible Stylesheet Language (XSL) und stellt eine Turing-vollständige Sprache dar.”

aus Wikipedia

XSLT - Skeleton 1 // Bitte erstellen

- **Einen neuen Ordner (xslt)**
- **Datei Lager.xml in diesen Ordner kopieren**
- **Beispiele.xslt, HTML.xslt und CSV.xslt anlegen**
- **Folgenden Gerüst in die Dateien kopieren**

```
<?xml version="1.0"?>
<xsl:stylesheet
    version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform "
    xmlns:tns="http://sobek-innovations.de/Waren ">


</xsl:stylesheet>
```

XSLT - Aus mehreren Standards bestehend

XSLT - Transformation von XML Dokumenten in andere

XPath - Ein Sprache zur Navigation in XML-Dokumenten

XSL-FO - Ein Sprache zur Formatierung von XML-Dokumenten (kein Bestandteil mehr)

XQuery - Eine Sprache zur Abfrage von Dokumenten (Anwendung von XPath auf mehrere Dokumente => doc("buecher.xml")/liste/buch/titel)

XSLT - <xsl:template>-Element

Das Element wird verwendet zum Aufbau von Template und damit Regeln wie Knoten transformiert werden sollen.

Das match Attribut definiert die ausgehenden Knoten, um das Template anzuwenden.

```
Beispiel // bitte erweitern:
```

```
<xsl:template match="/tns:waren">
  Nachfolgend finden Sie Information zu unserem Warenbestand für den Standort:

</xsl:template>
```

XSLT - <xsl:value-of>-Element

Diese Element extrahiert den Inhalt eines Elements aus dem XML-Quelldokument.

Dabei verwendet er das Attribut “select” welches dann den konkreten Inhalt im Element oder Attribut lokalisiert.

```
Beispiel // bitte erweitern:

<xsl:template match="/tns:waren">
  Nachfolgend finden Sie Information zu unserem Warenbestand für den Standort:
  <xsl:value-of select="./@standort">
</xsl:template>
```

XSLT - <xsl:for-each>-Element

Beispiel // bitte erweitern:

```
<xsl:template match="/">
  Nachfolgend finden Sie Information zu unserem Warenbestand für den Standort:
  <xsl:value-of select="./@standort">
  <xsl:for-each select="./ware">
    <xsl:value-of select="produktNummer"/>,
    <xsl:value-of select="regalNummer"/>,
     <xsl:value-of select="bestand"/>,
  </xsl:for-each>
</xsl:template>
```

XSLT Aufgabe 1

1. Bitte wandeln Sie den Lagerbestand in eine CSV Datei mit folgenden Format

```
produktNummer;regalNummer;bestand
1;10;10
2;20;1
```

XSLT Aufgabe 2

1. Bitte wandeln Sie den Lagerbestand in eine HTML Datei mit einer Tabelle.

```
<html>
    <head></head
    <body>
        <h1>Überschrift</h1>
        <table>
            <tr>
                <td>produktNummer</td>
                <td>regalNummer</td>
                <td>bestand</td>
            </tr>
            ...
        </table>
    </body>
</html>
```