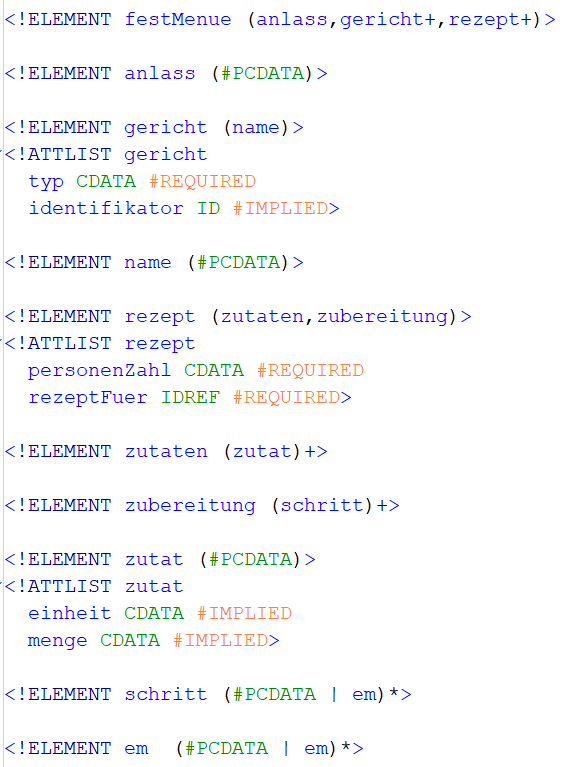
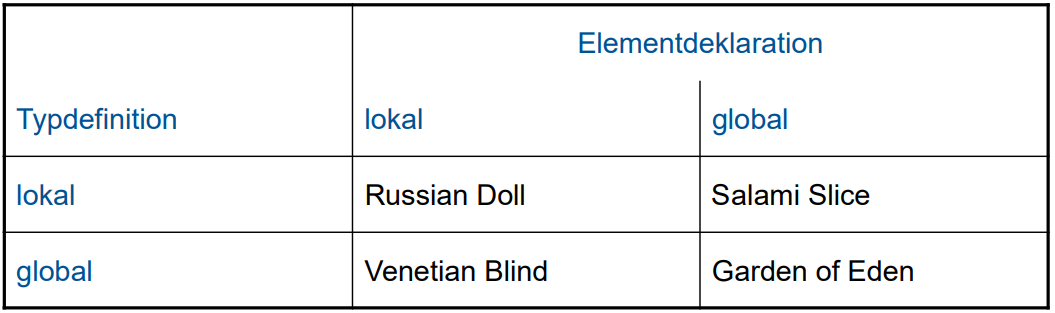
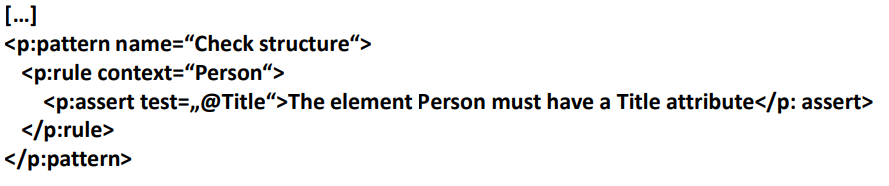
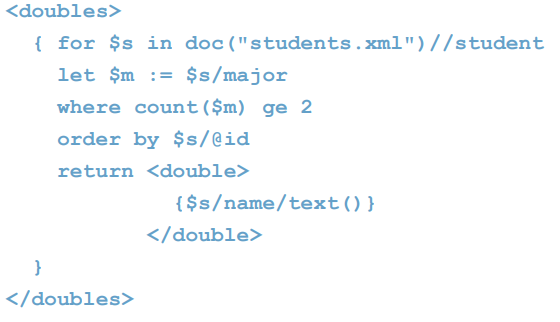
* **Document Engineering**: Elektronisches Publizieren, Document Management, Knowledge Management
* XML -> Sprache zur **Kodierung von** **strukturierten Dokumenten**
  + Standardisiertes Austauschformat für Dokumente
  + Zentrales Format - **Drehscheibencharakter**
  + Besteht aus: Elemente, Attribute, Text (Inhaltstext)
* Modell strukturierten Dokumente: **Trennung von Daten und Verwendungszweck**
* Für Präsentation in Ausgabemedium **Verarbeitungsvorschrift** (**Stylesheet**) benötigt
* **Metasprache** für Markupsprachen: XML stellt Regeln zur Verfügung, zum definieren von weiteren Metasprachen wie HTML
* **Entity-Referenzen**: definierte Kürzel zum Einfügen von Sonderzeichen die in XML eine andere Bedeutung haben, z.B. *&amp* für das *&* Zeichen
* **Parser**: Programme zur XML-Syntaxüberprüfung
  + Aufgaben: Syntaxüberprüfung, Überprüfen der Wohlgeformtheit
* **Unicode**: universeller Zeichensatz, Ziel: weltweiten Datenaustausch vereinfachen, zurzeit 136.690 Zeichen
* Zeichenkodierung: konkrete Zuordnung eines Codepoints zu einem Byte oder Bytesequenz
  + Beispiele: UTF-8 und UTF-16
* **UCS** (Universal Coded Character Set): 17 Ebenen zu 256 Kacheln à 256 Positionen (65.536)
* **Basic Multilingual Plane** (Plane 0): enthält hauptsächlich Schriftsysteme, die aktuell in Gebrauch sind
* **UTF** (Unicode Transformation Format): Zeichenkodierungen von Unicode
* **Surrogatpositionen:** platzsparendes Abspeichern auf Festplatten
* **Namespaces**: verschiedene Elemente einer Gruppe zuordnen, somit kann man **mehrere XML-Sprachen in einem Dokument mischen**, dazu da **Elemente** des XML **eindeutig** **zu** **identifizieren** und ggf. zu beschreiben und **Namenskonflikte zu vermeiden**
  + Standardisierte- und nutzerdefinierte Namensräume
* **qualifizierten Name**: Präfix + lokaler Name
* **erweiterter Name**: Namenraum-Name + lokaler Name
* **Datenmodelle**: legt fest, welche Bestandteile eines XML-Dokuments aus Sicht des Datenmodells wesentlich sind, abstrahiert von XML-Syntax, Klarheit und Präzision
  + **XML Information Set**: definiert Datenmodell in Form von Komponenten, Information Set entspricht Baumstruktur, Information Item entspricht Komponenten im Baum
  + **XDM**: basiert auf XML Infoset als konzeptuelle Sicht auf XML-Dokumente als Strukturen von Knoten (Element-, Attribut-, Text- und Namespaceknoten), XDM definiert ein Datenmodell für die Daten, auf denen **XPath, XQuery, XSLT** operieren
* Aspekte der Syntax von XML-Dokumenten, von denen XML Information Set abstrahiert:
  + XML-Deklaration, Begrenzer von Attributwerten, Kodierung von leeren Elementen, Leerzeichen in Tags, Reihenfolge der Attribute
* **Schemasprachen**: Festlegung von erlaubten Strukturen für eine Klasse von Dokumenten
  + Was wird festgelegt: Vorrat von Strukturelementen und Attributen, Vorschriften zu ihrer Kombination, Definition von Integritätsbedingungen
  + Beispiele: DTD, XML Schema, Relax NG, Schematron
* **XML Document Type Defintion (DTD):** Vokabular für **Elementnamen** und **Attributnamen**, Beziehung von Elementen und Attributen einschließlich **Identity Constraints** werden formal festgelegt, mit Hilfe von Elementdeklaration, Attributdeklaration
  + XML DTD mach XML zu **Metsprache für Markup-Sprachen**
  + Aufbau von XML DTD als **Katalog** (wie Salami-Slice)
  + Datentypen (vordefiniert, nicht erweiterbar):
    - Datentypen für Textinhalt für Elemente: *#PCDATA*
    - Datentypen für Attribute: *CDATA, ID, IDREF, IDREFS, ENTITY, ENTITIES, NAME*
  + Beispiel:

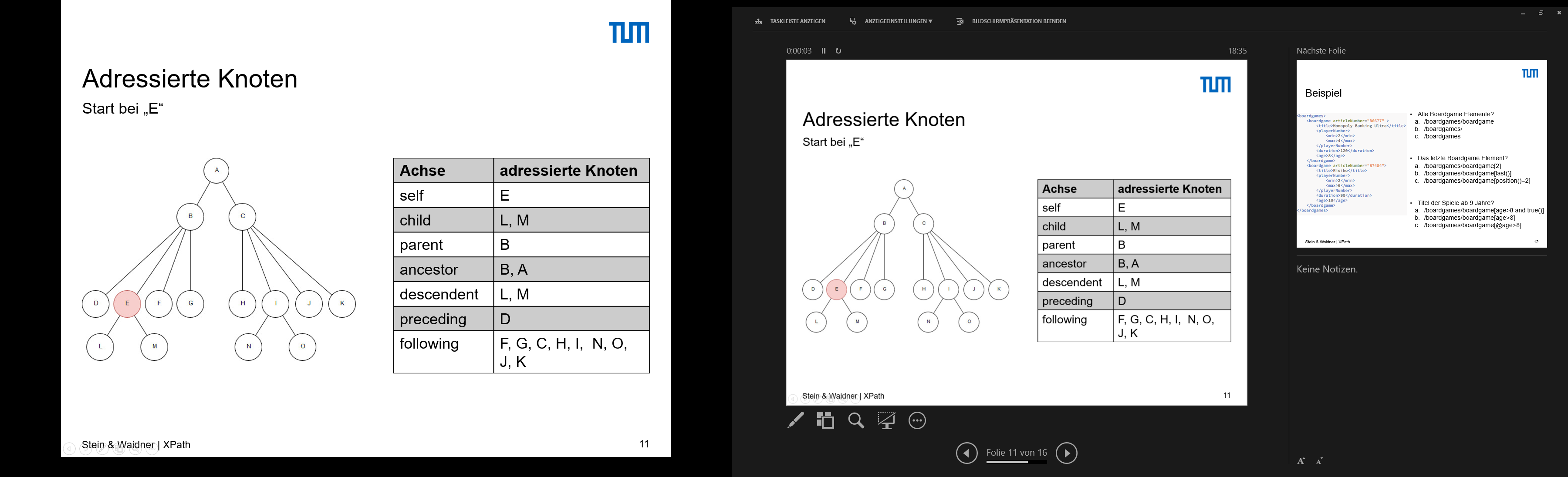


* **XML Schema**: konzeptuell komplex, mächtigere Beschreibungsmöglichkeit als DTD, adressiert als einzige Sprache folgende Aspekte: Datentypen (selber definieren), OO-Konzepte
  + **Datentyp** (SimpleType): Werte eines Datentyps sind bestimmte Zeichenketten: Strings mit bestimmten Bildungsgesetzen und Gleichheitsregelungen, Beispiele: xs:date, xs:integer, xs:string;
  + **3 Möglichkeiten Datentypen selbst zu definieren:**
    - **Restriction** (Facetten -> Beispiele: pattern, length, maxExclusive, enumeration), **List**, **Union**
  + **Strukturtyp** (ComplexType): Sequenz von Unterelementen, evtl. durchmischt mit Text oder reiner Inhaltstext, kombiniert mit Attributen
  + 
* **XML-Schematron**: regelbasiert, deklarativ (nicht generativ), komplementär zu XML Schema benutzbar, besonders gut zur Definition und Überprüfung von Identity Constraints geeignet, Aufbau aus Patterns (Gruppierung von Regeln), Regel gibt Kontext und Bedingung, die das Element erfüllen muss an (beides in Form eines XPath-Ausdrucks)
  + 
* **Wohlgeformtheit und Validität**:
  + **Wohlgeformt**: nur ein Wurzelelement, korrekte Klammerung, keine mehrfachen Attribute in Elemente
  + **Validität**: Referenzen definiert, Instanz erfüllt Deklaration von Elementen und Attributen, validierender Parser setzt Defaultwerte ein und erkennt Attributtypen wie ID, IDREF und ENTITY
* **Identity Constraints**: Eindeutigkeitsanforderungen zur **Identifizierung von Objekten** (Schlüssel), Integritätsanforderung zur **Referenzierung von Objekten** (Fremdschlüssel)
* XML Schema sieht vor: Datentypen **xs:ID und xs:IDREF**, Konzepte **cs:key und xs:keyref**
  + ID/IDREF: keine zusammengesetzten Schlüssel, nur ein Feld, keine separierbaren oder lokalen Gültigkeiten
  + key/keyref: erlaubt beliebig viele Attribute / Elemente zur Identifikation eines, Schlüsselwerte spezialisiert auf Schlüssel (keine globale Eindeutigkeit von IDs gefordert)
* **XQuery**: Abfragesprache für XML
  + Datenmodell **XDM** als Basis
  + Große Überlappung in der Funktionalität mit XSLT
  + Zentrales Sprachmittel: FLWOR-Ausdruck



* **XQuery Update Facility**: unterstützt lokale Änderungen von XML-Daten „am Platz“ Update-Primitive (delete node location, insert node items, replace node location, rename node location)
* **XQuery 3**.0: neue Klauseln für FLWOR-Statements (group by, windowing), Ausnahmebehandlung (try/catch), *sauberer* Modul-Import
* **XPath**: Teilsprache von **XQuery**, **Hauptfunktion**: Navigation durch die hierarchische Struktur von XML-Dokumenten, **verwendet in**: XML Schema Identity Constraints, Schematron, XSLT;
* **Location Path**: Teilsprache von XPath: **schrittweise Navigieren** durch Dokumentenbaum mit **Steps**
  + **Aufbau** Step:

**Achse::Knotentest [Prädikat] [Prädikat]**

* + Beispiel: **//p[@y='abc'][3]**
  + 
  + Ein XPath-Ausdruck wird von einem **Kontextknoten** ausgehend ausgewertet.
  + **Fallen** bei XPath: „unsichtbare“ Kindachse, Konvertierungsregeln nach Boolean in Prädikaten, Kontextnoten ändert sich dynamisch beim Auswerten von Teilausdrücken
* **XSLT (eXtensible Stylsheet Language Transformation)**: Programmiersprache deren Programme in XML kodiert sind, Hauptaufgabe: Transformation von XML-Dokumenten in andere Formate, wie HTML**, deklarativ** und wie XQuery **berechenbarkeitsuniversell,** XSLT macht XML **operationabel**
* **XSLT**-Prozessor führt Transformation durch, benötigt XML-Dokument und ein XML-Programm (**Stylesheet**), Stylesheet besitzt Basiskomponenten (**Templates**), die auch rekursiv aufgerufen werden können
* 3 **XSLT-Programmierstile**: Pull-Schema (**for each**, ein Template, typisch für datenorientierte Anwendung), Push-Schema (**apply template**, mehrere Templates, typisch für text-orientierte Anwendungen), funktionale Programmierung (typisch für komplexe Aufgaben, Turing-Vollständigkeit)
* Ein **XSL-FO-Prozessor** übersetzt ein **XSL-FO-Dokument** in ein **Präsentationsformat** (z.B. **PDF**). Der XSL-FO-Prozessor arbeitet dabei in **zwei Phasen** und **erzeugt** dabei ein **Zwischenformat**.
* Das Zwischenformat heißt **Area Tree**. Es enthält das Ergebnis der Formatierung in abstrakter Form, also eine genaue Beschreibung der Texte mit Fonts und Positionen auf den Seiten.
* Die zwei Phasen heißen **Formatierung** und **Rendering**.
  + Formatierung bestimmt Zeilen- und Seitenumbrüche, typographische Attribute für Schriften, Abstände etc. und hat den Area Tree als Ausgabe.
  + Rendering übersetzt den Area Tree in ein spezifisches Ausgabeformat, z.B. nach PDF, RTF oder GIF.
  + **Vorteil der Aufteilung** in 2 Phasen: Die aufwendige Aufgabe des Formatierens kann unabhängig vom Ausgabeformat realisiert werden und ist für die verschiedenen Ausgabeformate wiederverwendbar.
* Nennen Sie drei verschiedene Arten von Information, die in einem XML-Dokument, aber nicht in dem zugehörigen Baumdiagramm abgebildet sind.
  + XML-Deklaration
  + Entity-Referenzen
  + DTDs
  + Syntax-Details: Whitespace in Tags, Anführungszeichen um Attributwerte (einfach, doppelt), Leer-Elemente-Tags oder öffnendes und schließendes Tag ohne Inhalt