Vortrag gehalten am 20. Januar 2004
Otto-von-Guericke-Universität Magdeburg

XML
Document Object Model

Ein Vortrag von Markus Durzinsky

Student der Otto-von-Guericke-Universität Magdeburg

0. Gliederung

1. Modelle zum Einlesen von XML
2. DOM – Document Object Model
    1. Node-Schnittstelle
    2. spezialisierte Knoten
    3. andere Schnittstellen
3. JavaScript und DOM
4. Zusammenfassung
5. Quellenangabe

Modelle zum Einlesen von XML

- Einlesen von XML-Dokumenten als
  - reiner Text
  - Folge von Ereignissen
  - Baumstruktur

Ereignisorientiertes Modell

- spezifiziert in SAX (Simple API for XML)
- ein Beispiel:

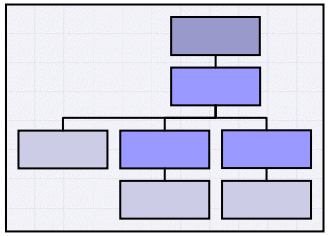
```
<person alter="25">
  <name>Peter Schmidt</name>
  <beruf>Student</beruf>
</person>
```

- attribut: alter, Wert: "25"
- startElement: person
- startElement: name
- content: "Peter Schmidt"
- endElement: name
- startElement: beruf
- content: "Student"
- endElement: beruf
- endElement: person

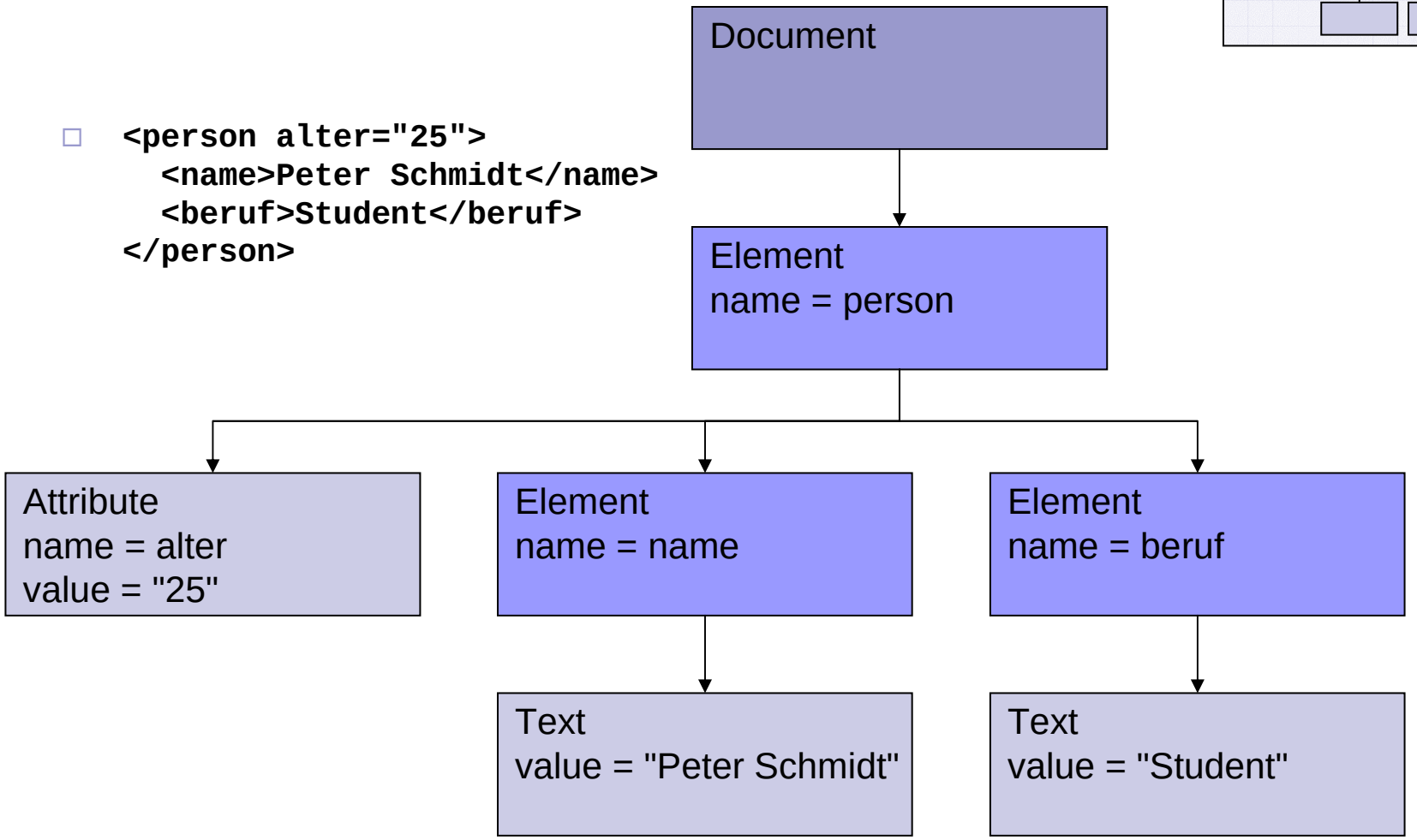
Baum-Modell

- wohlgeformte XML-Dokumente bilden Baumstrukturen
  - Ein Dokument besteht aus einem Element, dass Attribute, Text und Elemente enthält
  - Attribute und Text können auch als Knoten des Baumes betrachtet werden
- Spezifikation bildet DOM (Document Object Model)

Ein Beispiel



```
<person alter="25">
  <name>Peter Schmidt</name>
  <beruf>Student</beruf>
</person>
```



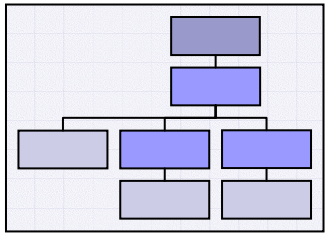
Vor- und Nachteile

- DOM
  - – lange "Wartezeit" beim Einlesen
  - – hoher Speicherbedarf
  - \+ beliebiger Zugriff auf Elemente
  - \+ Änderung möglich
- SAX
  - \+ Abarbeitung beginnt sofort
  - \+ kaum Speicherbedarf
  - – Dokument wird genau einmal durchlaufen
  - – nur Lesen

DOM – Document Object Model

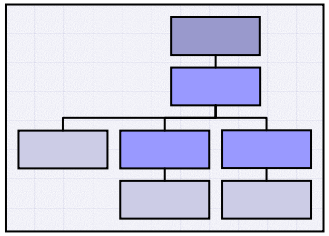
- Level-weise entwickelter Standard
- zur Beschreibung von Dokumenten in Baumstruktur
- Einlesen der Dokumente, Bearbeiten im Speicher
- Menge von objektorientierten APIs
  - generische Schnittstellen (**`Node`**, **`NodeList`**)
  - spezialisierte Schnittstellen von **`Node`**

Node-Schnittstelle



- Basisschnittstelle für alle Knoten
- Allgemeine Methoden zum Durchlaufen und Analysieren von Baumstrukturen
- Umwandlung in speziellen Knoten durch Casting
- nicht direkt erzeugbar (abstract)

Node-Schnittstelle



- Attribute
  - `DOMString nodeName`
  - `DOMString` ✍ `nodeValue`
  - `Short nodeType`
  - `Node parentNode`
  - `NodeList childNodes`
  - `Node firstChild`
  - `Node lastChild`
  - `Node previousSibling`
  - `Node nextSibling`
  - `NamedNodeMap attributes`
  - `Document ownerDocument`
  - `DOMString namespaceURI`
  - `DOMString` ✍ `prefix`
  - `DOMString localName`
- Methoden
  - `Boolean hasAttributes`
  - `Boolean hasChildNodes`
  - `Node insertBefore`
    - `Node newChild`
    - `Node refChild`
  - `Node replaceChild`
    - `Node newChild`
    - `Node oldChild`
  - `Node removeChild`
    - `Node oldChild`
  - `Node appendChild`
    - `Node newChild`

NodeList

- Zugriff auf geordnete Kinder eines Knotens

- Attribute
  - `Long length`
- Methoden
  - `Node item`
    - `Long index`

-
```
<person alter="25">
  <name>Peter Schmidt</name>
  <beruf>Student</beruf>
</person>
```

-
```
Node p = <Node person>
NodeList l = p.getChildNodes();
l.getLength(); -> 2
l.item(0);     -> <Node name>
l.item(1);     -> <Node beruf>
```

NamedNodeMap

- Zugriff auf ungeordneten Inhalt eines Knotens (i.A. Attribute)

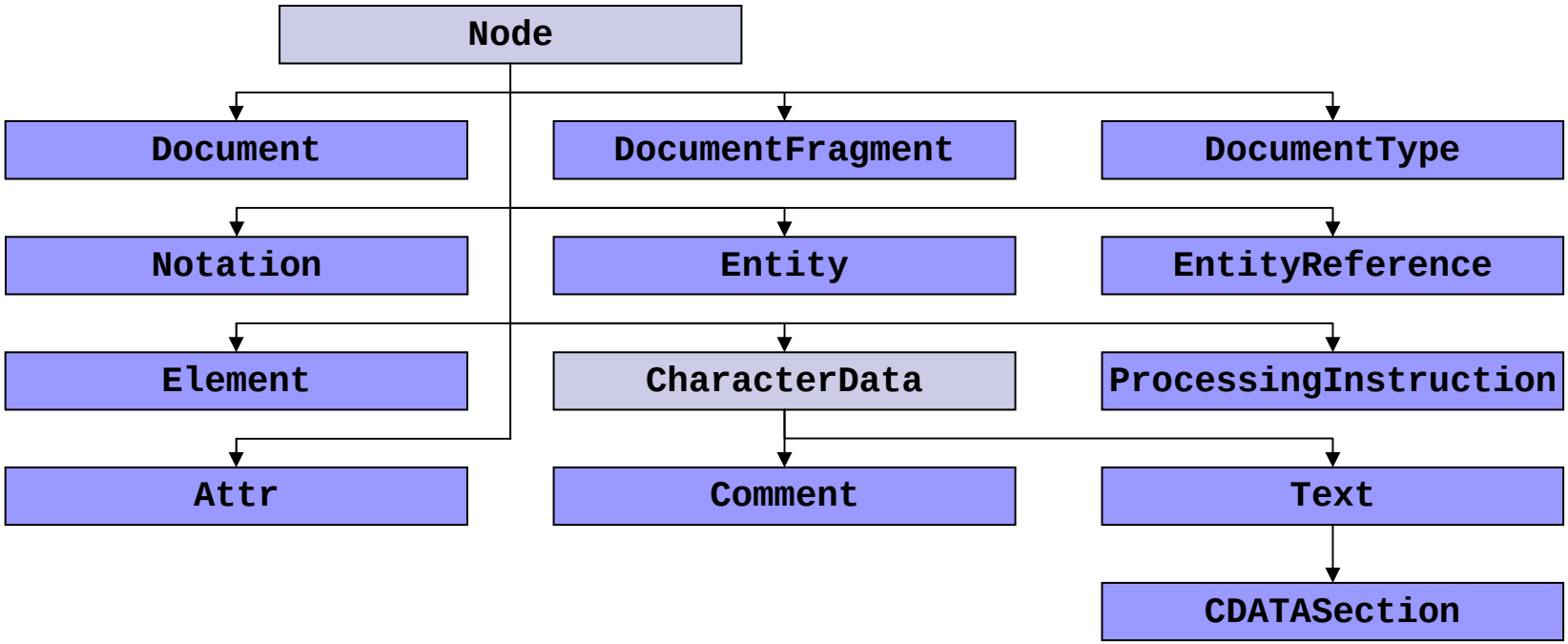
- Attribute
  - `Long length`
- Methoden
  - `Node item`
    - `Long index`
  - `Node getNamedItem`
    - `DOMString name`
  - `Node setNamedItem`
    - `DOMString arg`
  - `Node removeNamedItem`
    - `DOMString name`

```
<person alter="25">
  <name>Peter Schmidt</name>
  <beruf>Student</beruf>
</person>
```

```
Node p = <Node person>
NamedNodeMap l = p.getAttributes();
l.getLength(); -> 1
l.item(0);     -> <Attr alter>
l.getNamedItem("alter");
```

spezialisierte Schnittstellen

- erweitern **`Node`** durch Methoden und Attribute
- Behandlung von Dokument-spezifischen Aspekten

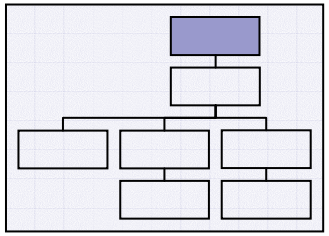


Werte für **nodeType**

- Knoten können in diese Schnittstellen gecastet werden

Wert	DOM-Schnittstelle
DOCUMENT_NODE	**Document**
DOCUMENT_FRAGMENT_NODE	**DocumentFragment**
DOCUMENT_TYPE_NODE	**DocumentType**
NOTATION_NODE	**Notation**
ENTITY_NODE	**Entity**
ENTITY_REFERENCE_NODE	**EntityReference**
PROCESSING_INSTRUCTION_NODE	**ProcessingInstruction**
ELEMENT_NODE	**Element**
ATTRIBUTE_NODE	**Attr**
COMMENT_NODE	**Comment**
TEXT_NODE	**Text**
CDATA_SECTION_NODE	**CDATASection**

`Document`-Schnittstelle



- von **`Node`** abgeleitet
- Wurzel des DOM-Baumes

- Attribute
  - **`DocumentType`** **`doctype`**
  - **`DOMImplementation`** **`implementation`**
  - **`Element`** **`documentElement`**
- Methoden
  - **`Node`** **`getElementById`**
    - **`DOMString`** **`elementId`**
  - **`NodeList`** **`getElementsByTagName`**
    - **`DOMString`** **`tagname`**
  - **`NodeList`** **`getElementByTagNameNS`**
    - **`DOMString`** **`namespaceURI`**
    - **`DOMString`** **`localName`**
  - Methoden zum Erstellen aller spezialisierter Knoten

`DocumentFragment`-Schnittst.

- von **`Node`** abgeleitet
- keine neuen Attribute und Methoden
- z.B. zum temporären Ablegen von Knoten

DocumentType-Schnittstelle

- von **Node** abgeleitet
- Keine Kindknoten
- Attribute
  - **NamedNodeMap entities**
  - **DOMString name**
  - **NamedNodeMap notations**
  - **DOMString publicId**
  - **DOMString systemId**

XML Ausschnitt:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
                      "DTD/xhtml1-transitional.dtd">
```

Notation-Schnittstelle

- von **Node** abgeleitet
- keine Kindknoten
- Attribute
  - **DOMString publicId**
  - **DOMString systemId**

XML Ausschnitt:

```
<!NOTATION jpeg SYSTEM "bild/jpeg">
```

`Entity`-Schnittstelle

- von **`Node`** abgeleitet
- befinden sich im **`DocumentType`**-Knoten
- Attribute
  - **`DOMString  notationName`**
  - **`DOMString  publicId`**
  - **`DOMString  systemId`**

XML Ausschnitt:

```
<!ENTITY mein_text "Hallo Welt">
<!ENTITY ein_bild SYSTEM "img/pic.jpeg" NDATA jpeg>
```

`EntityReference`-Schnittst.

- von **`Node`** abgeleitet
- keine neuen Attribute und Methoden
- Entityname in **`nodeName`** (aus **`Node`**)
- ein Parser muss diese Schnittstelle nicht benutzen

XML Ausschnitt:

```
&mein_text;
<bild quelle="ein_bild" />
```

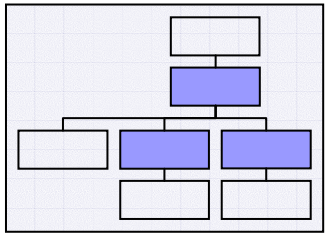
ProcessingInstruction

- von **Node** abgeleitet
- keine Kindknoten
- Attribute
  - **DOMString** ✍ **data**
  - **DOMString** **target**

XML Ausschnitt:

```
<?php
  echo 'Hello World';
?>
```

`Element`-Schnittstelle



- von **`Node`** abgeleitet
- bei XML am häufigsten verwendet

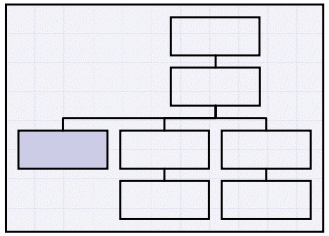
- Attribute
  - `DOMString tagName`
- Methoden
  - `Boolean hasAttribute`
    - `DOMString name`
  - `Attr getAttributeNode`
    - `DOMString name`
  - `Attr setAttributeNode`
    - `Attr newAttr`
  - `Attr removeAttributeNode`
    - `DOMString name`
  - `NodeList getElementsByTagName`
    - `DOMString name`

```
<person alter="25">
  <name>Peter Schmidt</name>
  <beruf>Student</beruf>
</person>
```

```
Node p = <Node person>
Element e = (Element) p;
e.hasAttribute("alter");
          -> true

e.getAttributeNode("alter");
          -> <Attr alter>
```

`Attr`-Schnittstelle



- von **`Node`** abgeleitet
- Enthält Text und EntityReferenzen als Kindknoten
- Attribute
  - **`DOMString name`**
  - **`DOMString value`**
  - **`Element ownerElement`**
  - **`Boolean specified`**

  XML Ausschnitt:

  `<img src="bild.gif" alt="ein Bild" />`

`CharacterData`-Schnittstelle

- von **`Node`** abgeleitet, abstract
- für Blöcke mit Zeichendaten

- Attribute
  - `DOMString` ✍ `data`
  - `Long` `length`
- Methoden
  - `Void` `appendData`
    - `DOMString` `arg`
  - `Void` `deleteData`
    - `Long` `offset`
    - `Long` `count`
  - `Void` `insertData`
    - `Long` `offset`
    - `DOMString` `arg`
  - `Void` `replaceData`
    - `Long` `offset`
    - `Long` `count`
    - `DOMString` `arg`

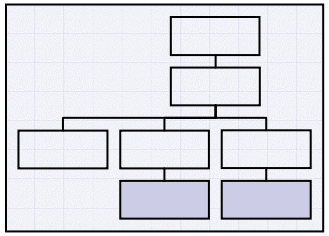
`Comment`-Schnittstelle

- von **`CharacterData`** abgeleitet
- keine Kindknoten
- keine neuen Attribute und Methoden
- Parser müssen diese Knoten nicht erstellen

XML Ausschnitt:

```
<!-- Ein Kommentar -->
```

`Text`-Schnittstelle



- von **`CharacterData`** abgeleitet
- keine Kindknoten
- kann in zwei Knoten geteilt werden
- Methoden
  - **`Text splitText`**
    - **`Long offset`**

`CDATASection`-Schnittstelle

- von **`Text`** abgeleitet
- keine Kindknoten
- keine neuen Attribute und Methoden

geerbte Attribute in spezialisierten Schnittstellen

Schnittstelle	**nodeName**	**nodeValue**	Attribute	Kinder
Document	**#document**	**-**	**-**	genau ein Kind
DocumentType	Typ	**-**	**-**	**-**
DocumentFragment	**#document-fragment**	**-**	**-**	beliebig viele
Entity	Entityname	**-**	**-**	Ersetzung
EntityReference	Entityname	**-**	**-**	Ersetzung
Notation	Notationname	**-**	**-**	**-**
ProcessingInstruction	Ziel	Inhalt	**-**	**-**
Element	Tag-Name	**-**	**NamedNodeMap**	Kindknoten
Attribut	Attributname	Attributwert	**-**	Inhalt als Text
Comment	**#comment**	Inhalt	**-**	**-**
Text	**#text**	Inhalt	**-**	**-**
CDATASection	**#cdata-section**	Inhalt	**-**	**-**

DOMImplementation

- Schnittstelle, mit der Dokumente erzeugt werden können

- Methoden
  - **Document createDocument**
    - **DOMString namespaceURI**
    - **DOMString qualifiedName**
    - **DocumentType doctype**
  - **DocumentType createDocumentType**
    - **DOMString qualifiedName**
    - **DOMString publicId**
    - **DOMString systemId**
  - **Boolean hasFeature**
    - **DOMString feature**
    - **DOMString version**

DOMParser

- keine offizielle Spezifikation
- Einlesen und Validieren von XML-Dateien
- Erstellen einer **`Document`**-Instanz
- Beispiel:

```
DOMParser dp = new DOMParser();
dp.parse("myfile.xml");

Document doc = dp.getDocument();
Node root    = doc.getDocumentElement();
```

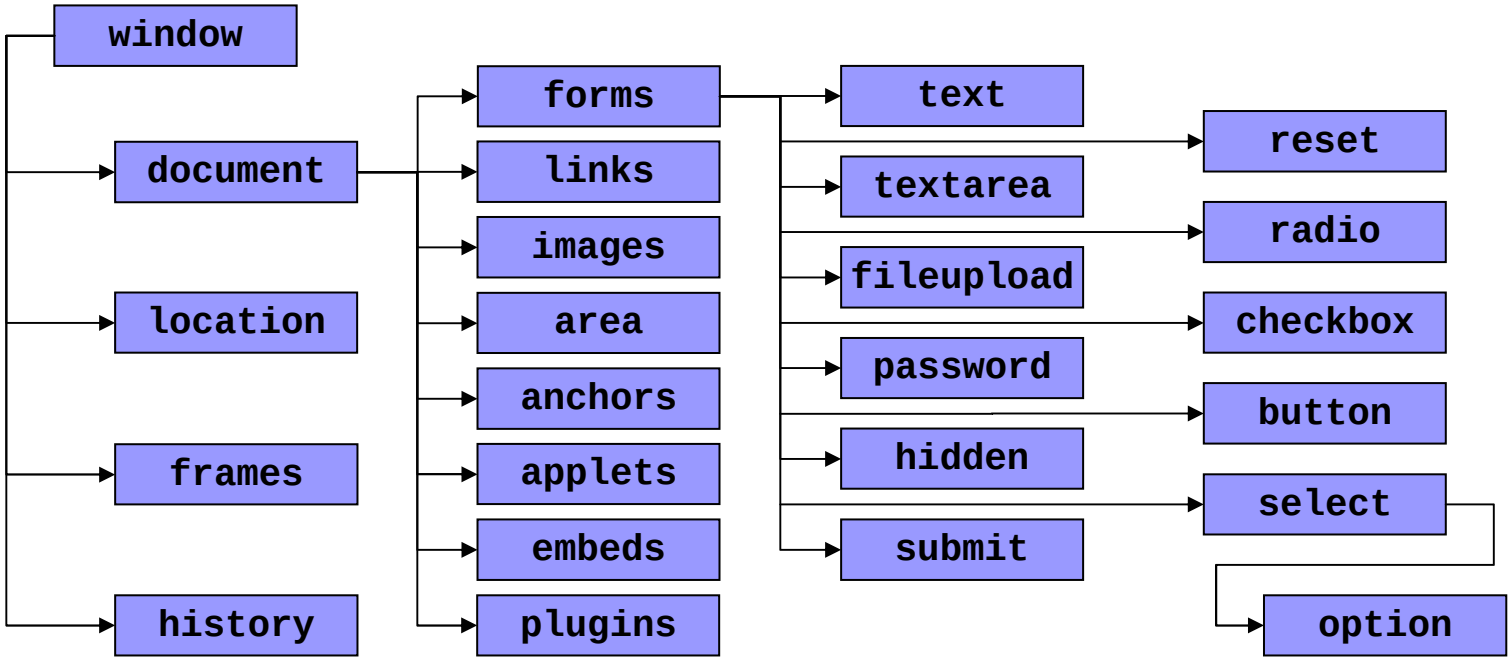
JavaScript und DOM

- DOM ist in JavaScript integriert
- Zugriff auf das Dokument direkt im Browser

```
<html>
<head><title>my first JavaScript</title></head>
<body>
<script language="JavaScript">
  document.writeln('Hello World!');
</script>
</body>
</html>
```

DOM-Struktur in JavaScript

- der Hierarchiebaum, der im Browser zur Verfügung steht:



Zugriff auf Kindknoten

- stark vereinfachter Zugriff
- Attribute von Elementen direkt als Attribute der Objekte
- Knotenliste
  - `window.frames[0]`
- Namen
  - `<any_tag id="foo"> oder <any_tag name="foo">`
  - `document.all.foo`
  - `document.getElementById("foo")`

Ändern von Attributen

- mouseover-Funktion bei Bildern

```
<script>
function picin() {
  document.pic.src = 'otto2.gif';
}

function picout() {
  document.pic.src = 'otto.gif';
}
</script>
```

```
<img name="pic" src="otto.gif"
     onmouseover="picin()" onmouseout="picout()">
```

Auflisten aller Knoten

- Durchlaufen aller Knoten des Dokumentes

```
<script>
function showElements() {
  var s = '';
  for (var i = 0;  i < document.all.length;  i++) {
    s = s + document.all.item(i).tagName + '\n';
  }
  document.myform.text2.value = s;
}
</script>
```

```
<form name="myform">
  <textarea rows="10" cols="40" name="text2">old text</textarea>
</form>
```

Hinzufügen von Knoten

- Text in einem neuen Abschnitt hinzufügen

```
<script>
function insertText() {
  var parent = document.createElement("p");
  var textNode = document.createTextNode("the text goes here");

  parent.appendChild(textNode);
  document.getElementById("foo").appendChild(parent);
}
</script>
```

```
<p name="foo">
  new text goes here:
</p>
```

Hinzufügen von Knoten

- ein Bild hinzufügen

```
<script>
function insertText() {
  var img = document.createElement("img");
  img.src = "otto.gif";

  document.getElementById("foo").appendChild(img);
}
</script>
```

```
<p name="foo">
  image goes here:
</p>
```

Zusammenfassung

- Was man aus diesem Vortrag erfahren sollte
  - DOM spezifiziert Datentypen zur Verarbeitung von Dokumenten in Baumstruktur
  - DOM legt das Bewegen durch den Baum und den Zugriff auf die Daten fest
  - Elemente aus XML-Dateien sind Knoten des DOM-Baumes
  - Attribute, Text und Unterelemente sind Kindknoten

Quellenangabe

- Elliotte Rusty Harold, W. Scott Means; XML in a nutshell; Deutsche Übersetzung der 2. Auflage 2003; O'Reilly Verlag GmbH & CO. KG
- Client-Side JavaScript Guide - Version 1.3; Netscape Communications Comporation 1999