FH Bielefeld
University of
Applied Sciences

Campus Minden

Studiengang Informatik Fachbereich Technik

FH Bielefeld
University of
Applied Sciences

Campus Minden

Webbasierte Anwendungen
SS 2015

Clientseitige Implementierungstechnologien: Document Object Model

Dozentin: Grit Behrens
mailto:grit.behrens@fh-bielefeld.de

Studiengang Informatik Fachbereich Technik

Lehrinhaltsübersicht der Vorlesungen zu WBA

1. Einführung in WBA
2. Wiederholung: Grundlagen des WWW, HTML und HTTP
3. **Clientseitige Implementierungstechnologien:** Javascript, **DOM**, Ajax, (Java-Applet)
4. Serverseitige Implementierungstechnologien: JSP, Java-Servlet
5. Anbindung von Datenbanken mit Java
6. WEB-Frameworks: (Struts), JSF

Clientseitige Implementierungs-technologien (II):DOM

1. Einführung
2. Der Dokumentenbaum
3. Eigenschaften von Knoten
4. Methoden
5. Zugriff auf Elemente
    1. Eigenschaften
    2. Elemente selektieren
    3. Attribute bearbeiten
    4. Elemente erzeugen
6. Zusammenfassung

Clientseitige Implementierungs-technologien (II):DOM

1. **Einführung**
2. Der Dokumentenbaum
3. Eigenschaften von Knoten
4. Methoden
5. Zugriff auf Elemente
    1. Eigenschaften
    2. Elemente selektieren
    3. Attribute bearbeiten
    4. Elemente erzeugen
6. Zusammenfassung

Einführung in das Document Object Model (DOM)

Bedeutung:

- ***Fundament*** moderner Webanwendungen
- Offizielle ***Standards des W3C***, die zum großen Teil in aktuellen Browsern gleich interpretiert werden
- Während der Browserkriege in erster Version entstanden und ***Meilenstein*** für Beginn einer neuen Generation des WWW

Technische Beschreibung:

- ***plattform- und sprachunabhängig***
- ***Schnittstelle*** für den ***Zugriff auf XML- und HTML-Dokumente***
- ***dynamischer Zugriff***: Bearbeitung und Löschen von einzelnen Dokumentelementen (Inhalt, Struktur, Layout)
- Bearbeitung des ***DOM mit JavaScript***

  (nach ECMA262 Standard siehe http://www.ecma-international.org/publications/standards/Ecma-262.htm

Einführung in das Document Object Model (DOM)

Standardisierungsprozess:

- seit 1998 ein Standard des W3C

DOM Level 0

- nicht formal spezifiziert.
- bezeichnet mittels JavaScript nutzbare Techniken zum ***Zugriff auf HTML-Dokumente***

DOM Level 1

- ***Bewegen im DOM-Baum*** und ***Manipulation der Knoten*** inklusive des ***Einfügens neuer Elemente*** und des Setzens von Attributen.

DOM Level 2

- XML-Namensraum-Unterstützung
- Ausweitung auf ***XHTML-Dokumente***
- ***dynamisches Auslesen, Hinzufügen und Ändern des Layouts*** über Cascading Style Sheets (CSS).
- ***Verarbeitung von Ereignissen im Dokument***, z.B. Benutzeraktionen.
- ***Durchlaufen des Knotenbaums*** anhand von bestimmten ***Auswahlkriterien***

DOM Level 3 (noch unterschiedlich in Browsern umgesetzt)

- verbesserte ***Ausnahmebehandlung*** und Umgang mit Zeichenkodierungen
- ***Parsen von XML-Dokumenten*** in Zeichenketten in Dokument-Objekten
- ***Versendung von XML-Dokumenten über HTTP*** ähnlich dem XMLHttp-Request

Clientseitige Implementierungs-technologien (II):DOM

1. Einführung
2. **Der Dokumentenbaum**
3. Eigenschaften von Knoten
4. Methoden
5. Zugriff auf Elemente
    1. Eigenschaften
    2. Elemente selektieren
    3. Attribute bearbeiten
    4. Elemente erzeugen
6. Zusammenfassung

Der Dokumentenbaum

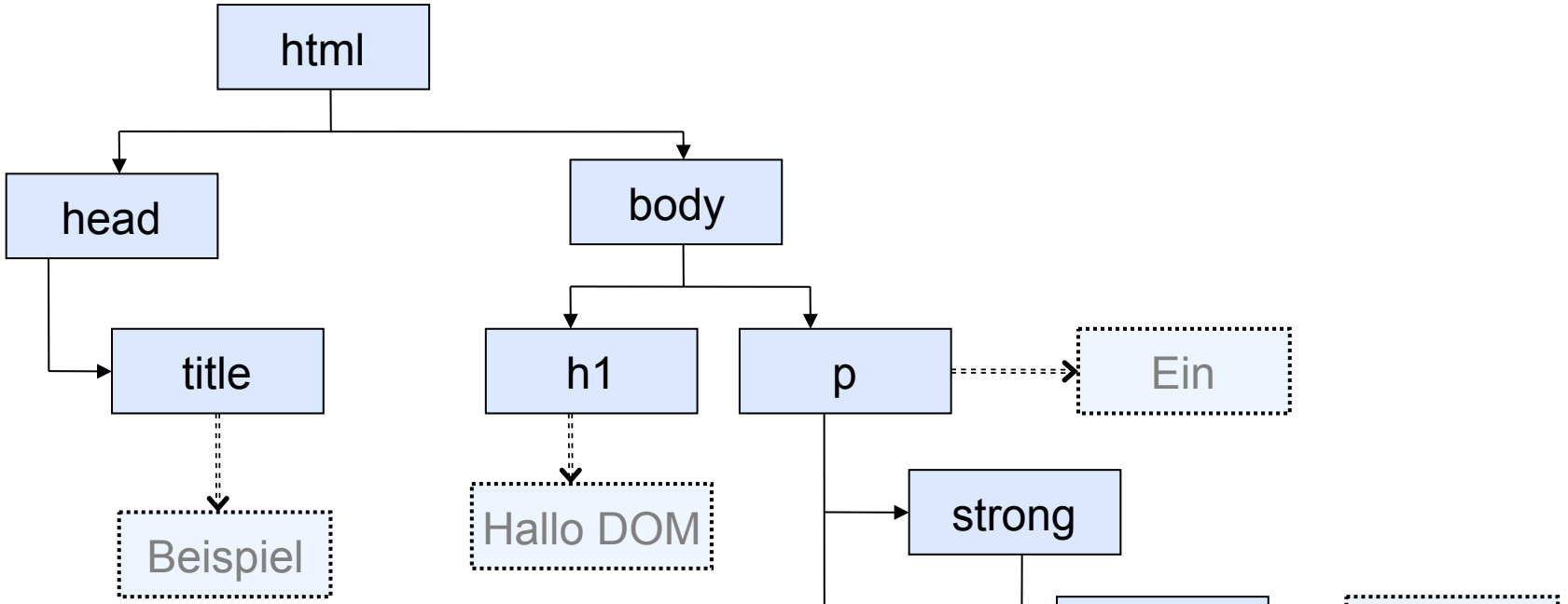
Eine vollständig übertragene Webseite repräsentiert gemäß dem W3C-DOM-Standard einen hierarchisch geordneten Dokumentenbaum, in dem jedes HTML-Element einen Knoten bildet.

Beispiel:

```
<html>
<head>
   <title>Beispiel</title>
</head>
<body>
<h1>Hallo DOM</h1>
<p>Ein <strong><em>kurzes</em></strong> Beispiel</p>
</body>
</html>
```

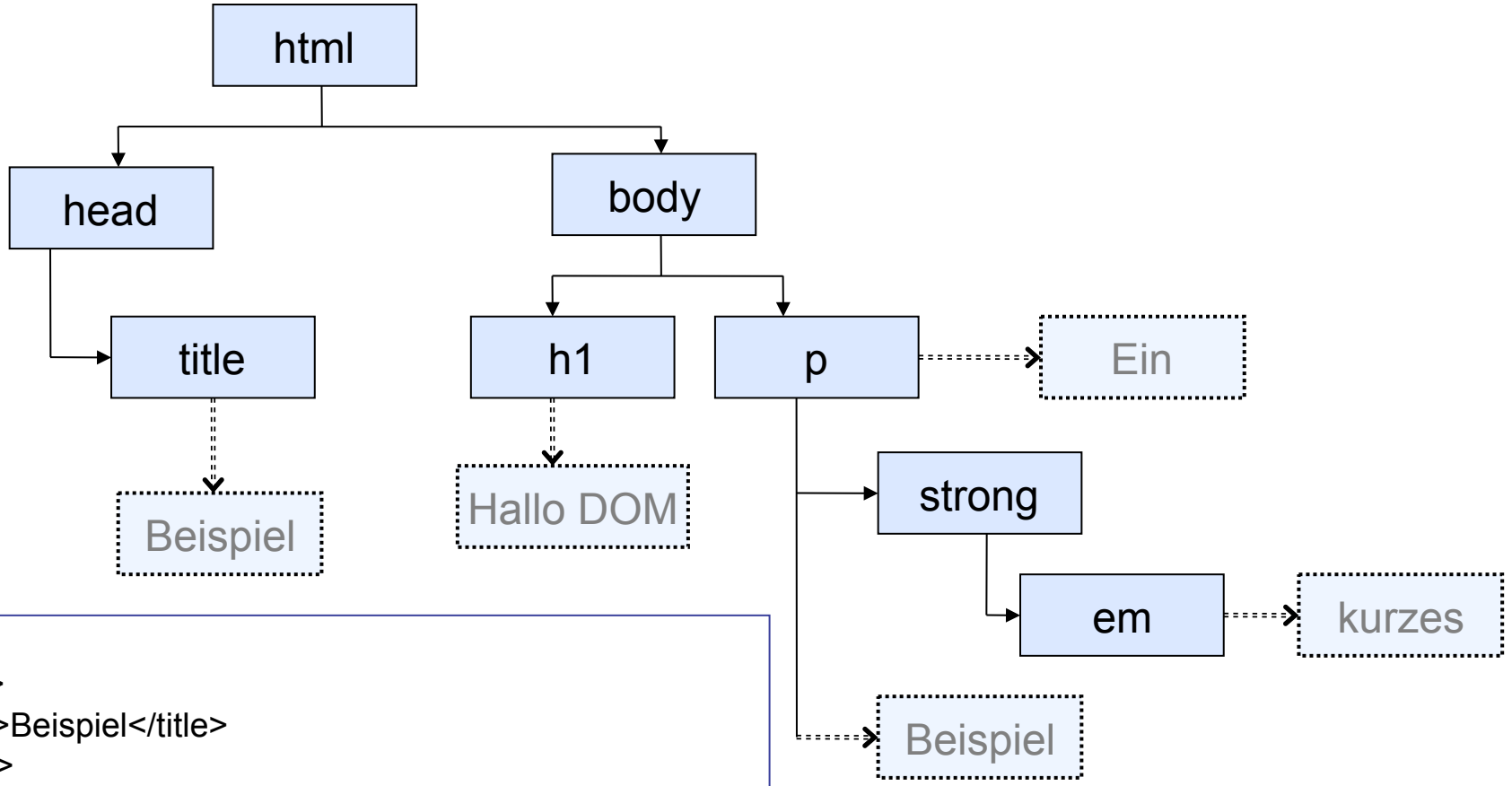
Aufgabe: Zeichnen Sie den hierarchisch geordneten Dokumentenbaum!

Der Dokumentenbaum




```
<html>
<head>
  <title>Beispiel</title>
</head>
<body>
<h1>Hallo DOM</h1>
<p>Ein <strong><em>kurzes</em></strong> Beispiel</
p>
</body>
</html>
```

Der Dokumentenbaum




```
<html>
<head>
  <title>Beispiel</title>
</head>
<body>
<h1>Hallo DOM</h1>
<p>Ein <strong><em>kurzes</em></strong> Beispiel</
p>
</body>
</html>
```

Knoten im Dokumentenbaum

```
<html>
<head>
  <title>Beispiel</title>
</head>
<body>
<h1>Hallo DOM</h1>
<p>Ein <strong><em>kurzes</em></strong> Beispiel</
p>
</body>
</html>
```

Aufgabe: Nennen Sie die Knoten im hierarchischen Dokumentenbaum!

html, head, title, body, h1, p, strong, em

Aufgabe: Geben Sie alle Eltern-/Kindbeziehungen innerhalb dieser Knotenmenge an!

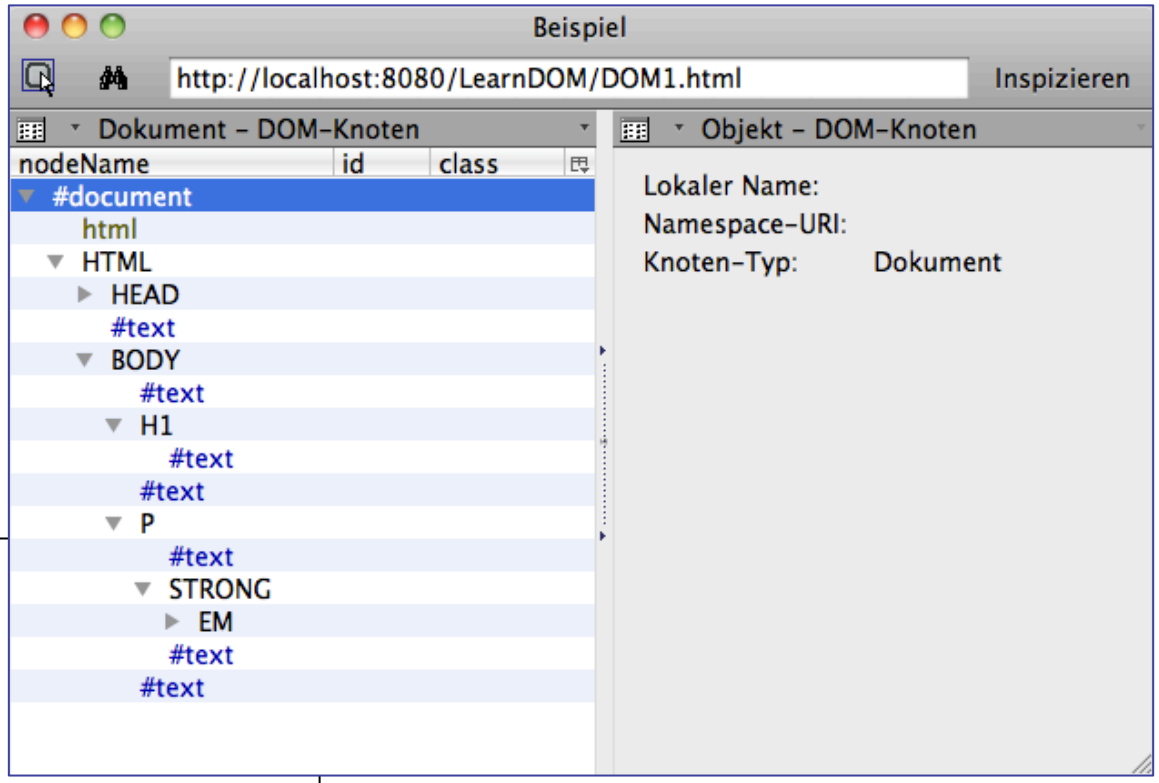
html -> head,
head -> title,
html-> body
body-> h1, p
p -> strong
strong ->em

Dokumentenbaum im DOM Inspector

Hilfreiches Tool:

Mozilla basierte Browser (hier z.B. Mozilla Firefox) enthalten einen DOM Inspector, der den Dokumentenbaum darstellt.

•Aktueller Firefox (April 2015):
>Extras > Webentwickler > DOM-Insepctor

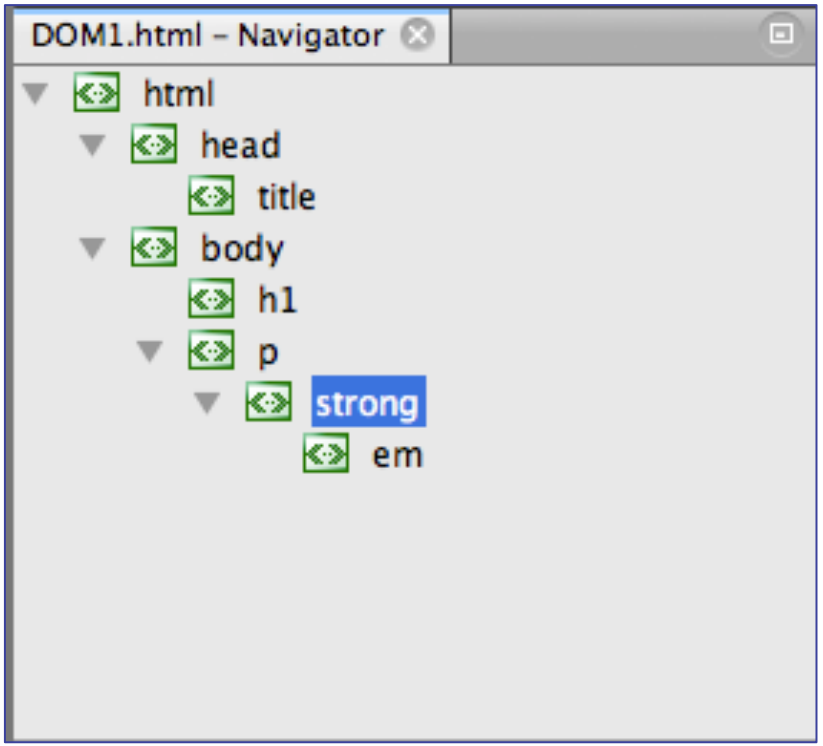



```
<html>
<head>
  <title>Beispiel</title>
</head>
<body>
<h1>Hallo DOM</h1>
<p>Ein <strong><em>kurzes</em></strong> Beispiel</
p>
</body>
</html>
```

DOM Navigator in SEU Netbeans

Hilfreiches Tool:

- kann in Netbeans SEU genutzt werden
- Optionswahl: -> Window -> Navigating -> Navigator



Knotentypen im Dokumentenbaum

Jeder Knoten besitzt Konstanten, Eigenschaften und Methoden, mit denen der Dokumentenbaum ausgelesen und bearbeitet werden kann.

Zum Beispiel besitzt die Eigenschaft **`nodeType`** als Wertebereich Konstanten:

`nodeType`	**Konstante**	**Beschreibung**
1	ELEMENT_NODE	Element
2	ATTRIBUTE_NODE	Attribut
3	TEXT_NODE	Text
8	COMMENT_NODE	Kommentar
9	DOCUMENT_NODE	Dokument

Beispielhafte Abfrage der konstanten Eigenschaft:

```
if (Knoten.nodeType == 8) alert(„Ich bin ein Kommentar“);
```

Clientseitige Implementierungs-technologien (II):DOM

1. Einführung
2. Der Dokumentenbaum
3. **Eigenschaften von Knoten**
4. Methoden
5. Zugriff auf Elemente
    1. Eigenschaften
    2. Elemente selektieren
    3. Attribute bearbeiten
    4. Elemente erzeugen
6. Zusammenfassung

Eigenschaften von Knoten im Dokumentenbaum

1. childNodes[], firstChild, lastChild

- **firstChild** zeigt auf den ersten Knoten der Struktur
- **lastChild** zeigt auf den letzten Knoten der Struktur
- **childNodes[]** Array ermöglicht Zugriff auf alle Elemente, Zählung beginnt mit 0
- Bei tiefer verschachtelten Knoten muss ggf. rekursiv durch alle Ebenen gegangen werden
  z.B. document.body.childNodes[i].childNodes[j].firstChild

2. nodeName, nodeValue, nodeType

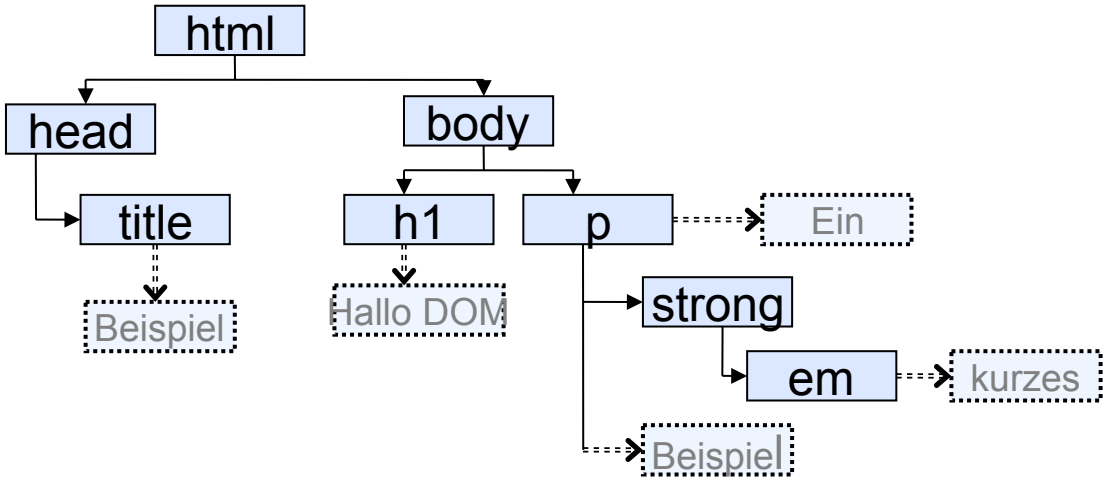
- **nodeValue** Textinhalt des Knotens, falls vom Typ TEXT_NODE, sonst **null**
- **nodeName** Namen des XHTML-Elements, bei TEXT_NODE : #text
- **nodeType** konstante Werte

Eigenschaften von Knoten im Dokumentenbaum

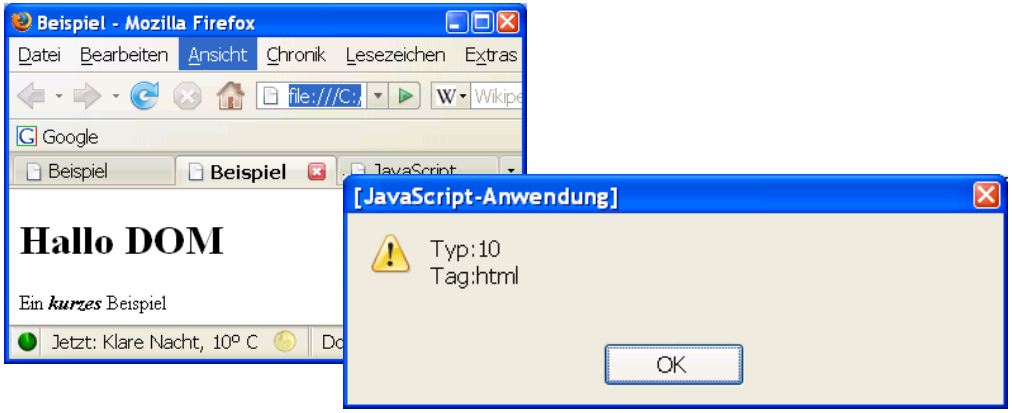
Beispiel:

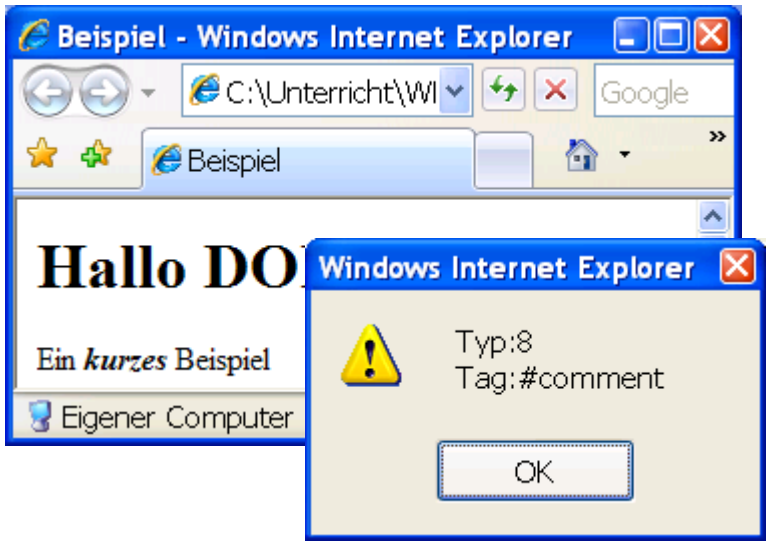
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de" lang="de">
<head>
  <title>Beispiel</title>
<script type="text/javascript">
//<![CDATA[
window.onload = function()
{
alert("Typ:"+document.firstChild.nodeType+"\n"+
     "Tag:"+document.firstChild.nodeName);
}
//]]>
</script>
</head>
<body><h1>Hallo DOM</h1>
<p>Ein <strong><em>kurzes</em></strong> Beispiel</p>
</body>
</html>
```

Eigenschaften von Knoten im Dokumentenbaum

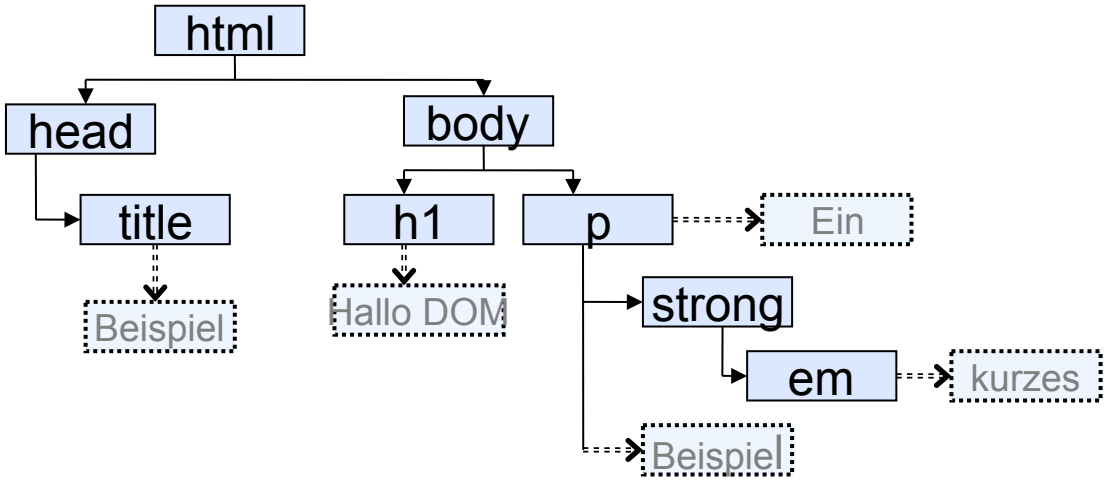


- `document.firstChild` zeigt im Beispiel auf Element „HTML“ (Firefox)
- andere Implementierung in MSIE





Eigenschaften von Knoten im Dokumentenbaum



Welche Ausgaben werden mit folgenden Eigenschaften erzeugt?

```
document.body.childNodes[0].nodeName
document.body.childNodes[1].nodeName
document.body.firstChild.nodeName
document.body.lastChild.nodeName
```

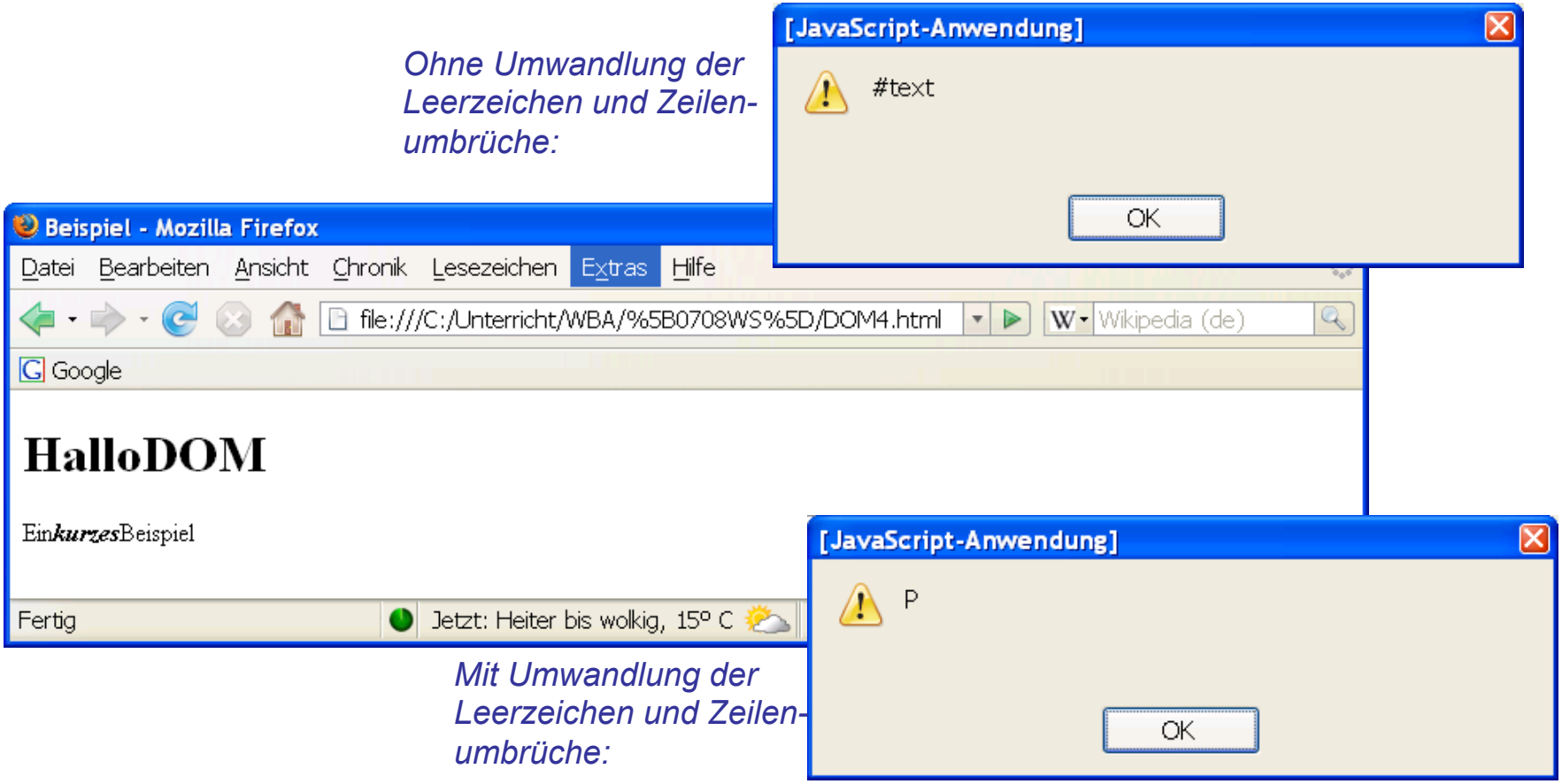
-> h1
-> p
-> h1
-> p

Code – Beispiel für Umwandlung der Leerzeichen und Zeilenumbrüche:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de" lang="de">
<head>
  <title>Beispiel</title>
<script type="text/javascript">
window.onload = function()
{
/*document.body.innerHTML = document.body.innerHTML.replace(/[\r\n|\s]/g,'');*/
//Umwandlung der Leerzeichen und Zeilenumbrüche für MF
alert(document.body.childNodes[document.body.childNodes.length-1].nodeName);
}
</script>
</head>
<body><h1>Hallo DOM</h1>
<p>Ein <strong><em>kurzes</em></strong> Beispiel</p>
</body>
</html>
```

Code - Beispiel:

Ohne Umwandlung der Leerzeichen und Zeilen-umbrüche:

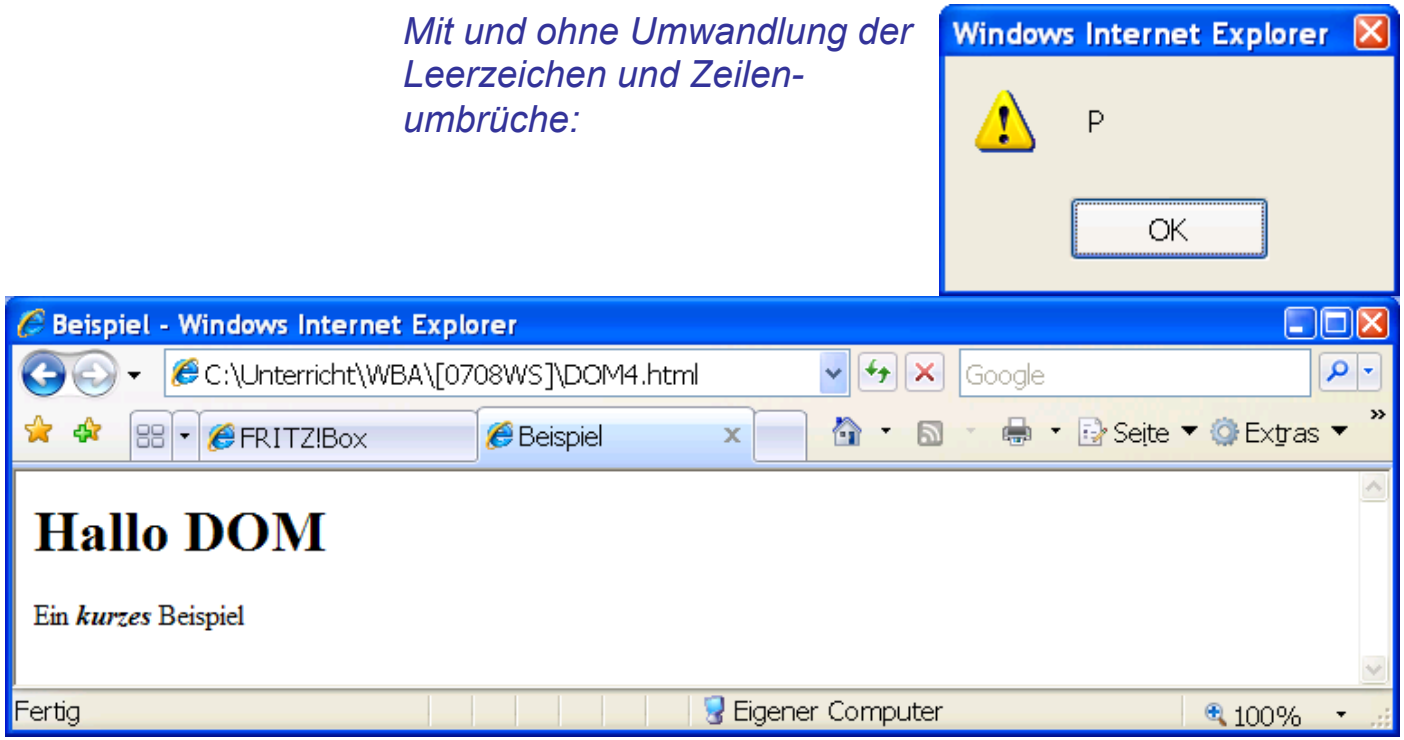


Mit Umwandlung der Leerzeichen und Zeilen-umbrüche:

Im firefox, die Ausgabe einmal mit Umwandlung `document.body.innerHTML = document.body.innerHTML.replace(/[\r\n|\s]/g,'')` , einmal ohne

Code - Beispiel:

Mit und ohne Umwandlung der Leerzeichen und Zeilen-umbrüche:



Im MSIE braucht die Umwandlung nicht durchgeführt zu werden.

Eigenschaften von Knoten im Dokumentenbaum

3. previousSibling, nextSibling (Sibling – Geschwister)

- Für horizontales Zugreifen im Dokumentenbaum
- `previousSibling` Knoten links vom aufrufenden Element
- `nextSibling` Knoten rechts vom aufrufenden Element

Welche Ausgaben erzeugen folgende Zeilen für unser Codebeispiel?

```
alert(document.body.childNodes[0].nextSibling.nodeName)
alert(document.body.firstChild.nextSibling.nodeName)
alert(document.body.childNodes[0].previousSibling.nodeName)
alert(document.body.firstChild.previousSibling.nodeName)
```

->p
->p
->null
->null

Hinweis: In den modernen Browsern werden bei dem Wert „null“ gar keine Alert-Fenster mehr angezeigt.

Eigenschaften von Knoten im Dokumentenbaum

4. parentNode, ownerDokument, attributes

- `parentNode` enthält Informationen über den Elternknoten, ggf. „null“
  für die Typen `DOCUMENT_NODE` und `ATTRIBUTE_NODE` sind keine Elternknoten vorhanden
- `ownerDokument` enthält das `dokument`-Objekt, zu dem dieser Knoten gehört
- `attributes` Array mit allen Attributen eines Elements

Welche Ausgabe erzeugt die folgende Zeile für unser Codebeispiel?

```
alert(document.body.firstChild.parentNode.nodeName);
```

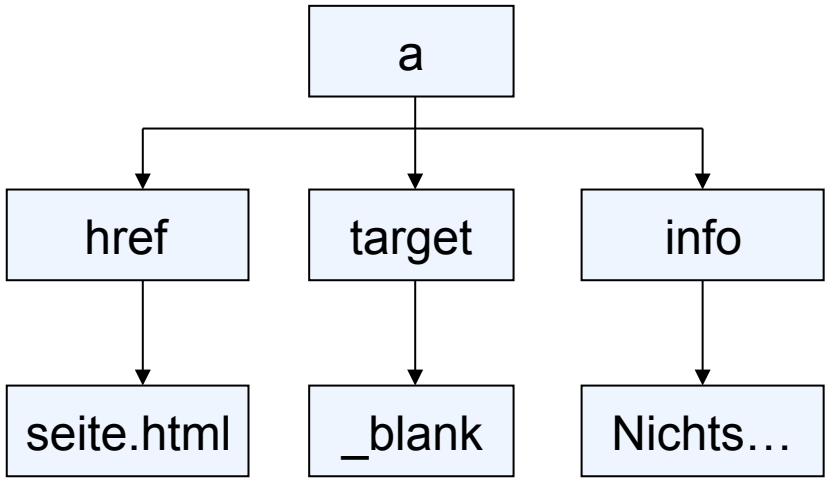
Eigenschaften der Knoten: Attribute

In DOM können alle in XHTML spezifizierten Attribute verwendet werden und es können eigene Attribute eingefügt werden.

Beispiel:

```
<a href="seite.html" target="_blank" info="Nichts besonderes.">Eine Seite</a>
```

- `href` und `target` sind Attributelemente des übergeordneten Elementknotens `a`
- `info` ist ein zusätzliches Attribut



Struktur für die Attribute im Beispiel eines Links

Codebeispiel für Attribute im DOM

Beispiel:

```
<a href="seite.html" target="_blank" info="Nichts besonderes.">Eine Seite</a>
```

- `href` und `target` sind Attributelemente des übergeordneten Elementknotens `a`
- `info` ist ein zusätzliches Attribut

Hinweis:
Codebeispiel funktioniert nur im MSIE, nicht im Firefox und auch nur, wenn die dtd **Nicht** erweitert ist… (DOM6.html)

Codebeispiel für Attribute im DOM

Beispiel:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de" lang="de">
<head>
   <title>JavaScript und DOM</title>
<script type="text/javascript">
window.onload = function()
{
    /*document.body.innerHTML = document.body.innerHTML.replace(/[\r\n|\s]/
g,''); */
    alert(document.body.firstChild.attributes.notiz.nodeValue);
}
</script>
</head>
<body>
<h1 notiz="Eine Überschrift">Hallo DOM</h1>
<p>Ein <strong><em>kurzes</em></strong> Beispiel</p>
</body>
</html>
```

Clientseitige Implementierungs-technologien (II):DOM

1. Einführung
2. Der Dokumentenbaum
3. Eigenschaften von Knoten
4. **Methoden**
5. Zugriff auf Elemente
    1. Eigenschaften
    2. Elemente selektieren
    3. Attribute bearbeiten
    4. Elemente erzeugen
6. Zusammenfassung

Methoden im DOM

Die Methoden im DOM sind mächtige Werkzeuge bei der Bearbeitung einer Seite. Es können dynamisch Inhalte gelöscht, verändert oder eingestellt werden. Mit den Methoden können nützliche Informationen für die Webanwendung abgerufen werden.

Methode	Argument	Beschreibung
appendChild	newNode	fügt Knoten an das Ende ein.
cloneNode		kopiert einen Knoten
hasAttributes		prüft, ob ein Knoten Attribute besitzt
hasChildNodes		prüft, ob ein Knoten weitere Kindelemente besitzt
insertBefore	newNode, target	fügt einen Knoten vor einem anderen Knoten ein
removeChild		entfernt ein Element
replaceChild	newNode, oldNode	Ersetzt ein Element des Dokumentenbaumes durch ein anderes Element

Übersicht über die wichtigsten Methoden

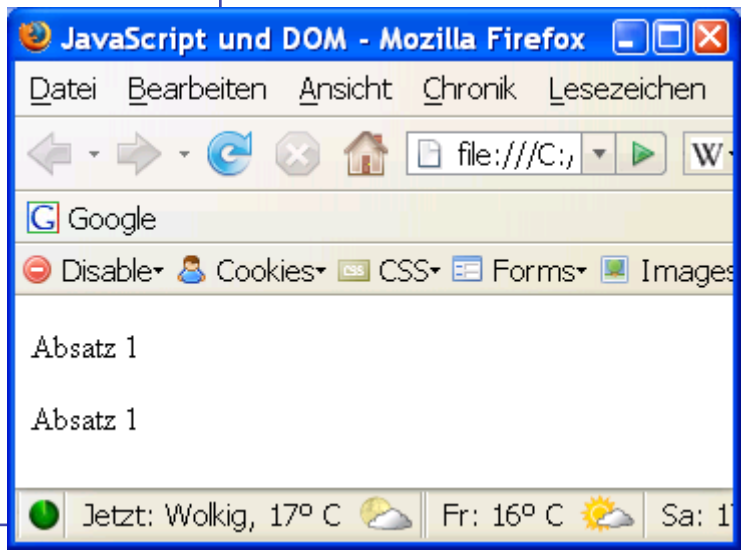
Beispiel: `appendChild()`

Frage:
Welche Ausgabe wird durch das nachfolgende Beispiel erzeugt und warum?

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de" lang="de">
<head>
            <title>JavaScript und DOM</title>
</head>
<body><p>Absatz 1</p>

<script type="text/javascript">
//<![CDATA[
var p = document.body.firstChild.cloneNode(true);
document.body.appendChild(p);
//]]>
</script>

</body>
</html>
```

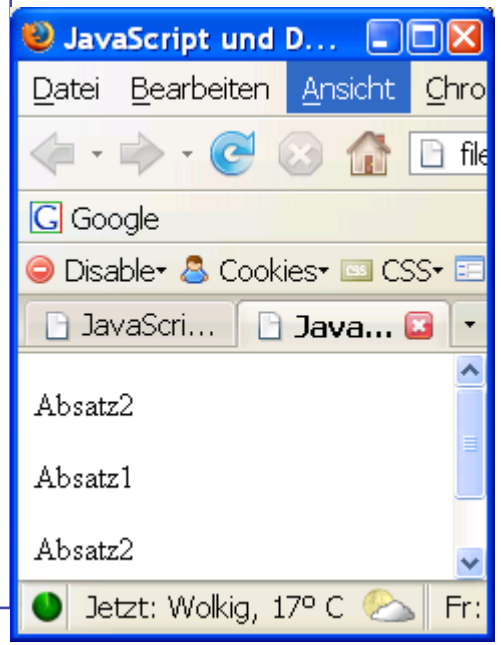


Antwort:
Die Kopie des Knotens im body (`document.body.firstChild.cloneNode(true)`) wird im Dokument an das Ende angehängt (`document.body.appendChild(p)`).

Beispiel: `insertBefore()`

Frage:
Welche Ausgabe wird durch das nachfolgende Beispiel erzeugt und warum?

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de" lang="de">
<head>
            <title>JavaScript und DOM</title>
</head>
<body>
<p>Absatz 1</p>
<p>Absatz 2</p>
<script type="text/javascript">
//<![CDATA[
document.body.innerHTML = document.body.innerHTML.replace(/[\r\n|\s]/g,'');
var node = document.body.firstChild;
var p2 = document.body.childNodes[1].cloneNode(true);
document.body.insertBefore(p2,node);
//]]>
</script>
</body>
</html>
```



Antwort:
Die Kopie des zweiten Knotens „p2“ im body (`document.body.childNodes[1].cloneNode(true)`) wird vor das erste Element „node“ gehängt (`document.body.insertBefore(p2,node)`).

Beispiel: `removeChild()`

Frage:
Welche Ausgabe wird durch das nachfolgende Beispiel erzeugt und warum?

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de" lang="de">
<body>
<p>Zeile 1</p>
<p>Zeile 2</p>
<script type="text/javascript">
//<![CDATA[
document.body.innerHTML = document.body.innerHTML.replace(/[\r\n|\s]/g,'');
var node = document.body.firstChild;
var temp = document.body.removeChild(node);
function undo()
{
   var node = document.body.firstChild;
   document.body.insertBefore(temp,node);
}
//]]>
</script>
<a href="#" onclick="undo();">Undo</a>
</body>
</html>
```

Beispiel: `removeChild()`

Antwort:
Löschen des ersten Elements und danach wieder rückgängig machen.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de" lang="de">
<body>
<p>Zeile 1</p>
<p>Zeile 2</p>
<script type="text/javascript">
//<![CDATA[
document.body.innerHTML = document.body.innerHTML.replace(/[\r\n|\s]/g,'');
var node = document.body.firstChild;
var temp = document.body.removeChild(node);
function undo()
{
   var node = document.body.firstChild;
   document.body.insertBefore(temp,node);
}
//]]>
</script>
<a href="#" onclick="undo();">Undo</a>
</body>
</html>
```

Beispiel: `replaceChild()`

Frage:

Welche Ausgabe wird durch das nachfolgende Beispiel erzeugt und warum?

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de" lang="de">
<body>
<p>Zeile 1</p>
<p>Zeile 2</p>
<script type="text/javascript">
//<![CDATA[
document.body.innerHTML = document.body.innerHTML.replace(/[\r\n|\s]/g,'');
var p1  = document.body.childNodes[0];
var p2  = document.body.childNodes[1];
var pc1 = p1.cloneNode(true);
var pc2 = p2.cloneNode(true);
document.body.replaceChild(pc2,p1);
document.body.replaceChild(pc1,p2);
//]]>
</script>
</body>
</html>
```

Antwort:

Die Position von zwei Absätzen wird im Dokumentenbaum ausgetauscht. Es wird dazu vorher jeweils eine Kopie des alten Knotens angefertigt.

Clientseitige Implementierungs-technologien (II):DOM

1. Einführung
2. Der Dokumentenbaum
3. Eigenschaften von Knoten
4. Methoden
5. **Zugriff auf Elemente**
    1. Eigenschaften
    2. Elemente selektieren
    3. Attribute bearbeiten
    4. Elemente erzeugen
6. Zusammenfassung

Zugriff auf einzelne Elemente

Es gibt eine Reihe von Eigenschaften und Methoden für den Zugriff und die Anwendung auf einzelne Elemente des Dokumentenbaumes.
Die Auswahl des Elements ist auch gezielt über die CSS-ID möglich mit der Methode `getElementById()`.

Eigenschaften des Document-Objekts

Eigenschaft des Document-Objekts	Beschreibung
`defaultView`	beschreibt die Standardansicht des aktuellen Dokuments, welche in Webbrowsern durch das **`window`**- Objekt repräsentiert wird
`doctype`	zeigt den Inhalt einer **`<!DOCTYPE ...>`** - Deklaration an, falls diese vorhanden ist, sonst wird **`null`** ausgegeben; nur lesender Zugriff möglich
`documentElement`	gibt die Referenz auf das Wurzelelement des Dokumentenbaumes
`styleSheets`	liefert ein Array mit Referenzen auf StyleSheets, die mittels des **`link`**- oder **`style`**-Tags in ein Dokument eingebunden sind.

Zugriff auf einzelne Elemente

Es gibt eine Reihe von Eigenschaften und Methoden für den Zugriff und die Anwendung auf einzelne Elemente des Dokumentenbaumes.
Die Auswahl des Elements ist auch gezielt über die CSS-ID möglich mit der Methode `getElementById()`.

Eigenschaft des Document-Objekts	Beschreibung
`defaultView`	beschreibt die Standardansicht des aktuellen Dokuments, welche in Webbrowsern durch das **`window`**- Objekt repräsentiert wird
`doctype`	zeigt den Inhalt einer **`<!DOCTYPE ...>`** - Deklaration an, falls diese vorhanden ist, sonst wird **`null`** ausgegeben; nur lesender Zugriff möglich
`documentElement`	gibt die Referenz auf das Wurzelelement des Dokumentenbaumes
`implementation`	DOMImplementation-Objekt, mit dem das aktuelle Dokument dargestellt wird
`styleSheets`	liefert ein Array mit Referenzen auf StyleSheets, die mittels des **`link`**- oder **`style`**-Tags in ein Dokument eingebunden sind.

Clientseitige Implementierungs-technologien (II):DOM

1. Einführung
2. Der Dokumentenbaum
3. Eigenschaften von Knoten
4. Methoden
5. Zugriff auf Elemente
    1. **Eigenschaften**
    2. Elemente selektieren
    3. Attribute bearbeiten
    4. Elemente erzeugen
6. Zusammenfassung

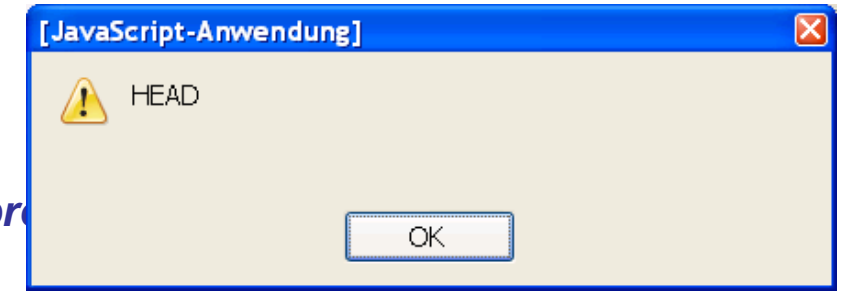
Eigenschaften des Dokument - Objektes

documentElement

- beinhaltet Referenz auf den Wurzelknoten des Dokumentenbaumes.
- in xhtml-Dokumenten zeigt es immer auf das **html**-Tag.
- wird häufig zur Auswertung von xml-Daten genutzt.

Beispiel:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de" lang="de">
<head>
        <title>JavaScript und DOM</title>
</head>
<body>
<script type="text/javascript">
alert(document.documentElement.childNodes[0].nodeName);
</script>
</body>
</html>
```



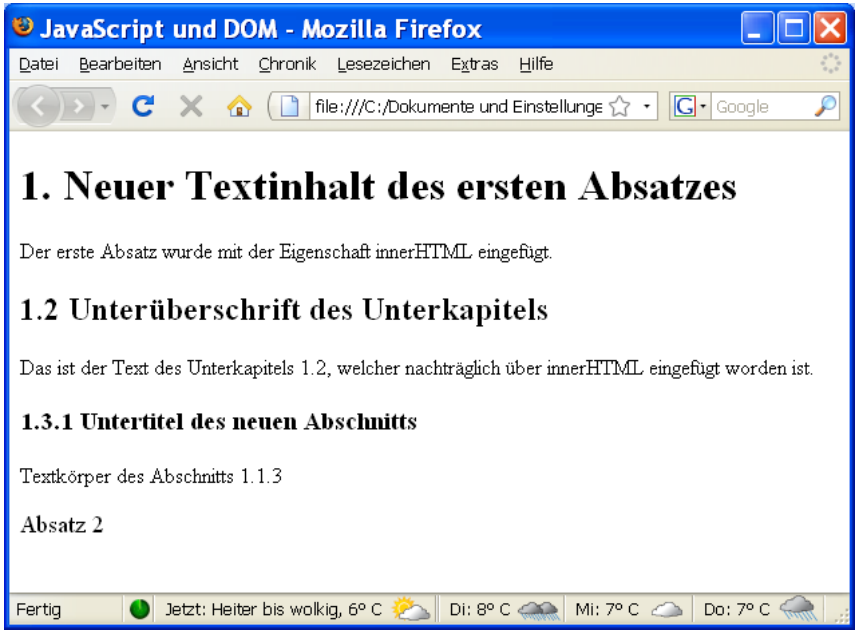
Frage: Welche Ausgabe wird von dem Beispielpr

Antwort: es wird der Name des ersten Kind-Elements in einem Alert-Fenster geliefert.-> **head**

Eigenschaft : `innerHTML`

- erlaubt Auslesen und Ändern eines Elementknotens
- ist aktuell nicht Bestandteil der DOM-Spezifikation
- wurde von Microsoft mit MSIE 5.0 eingeführt, später vom Netscape Browser 6.0 übernommen und hat sich stark verbreitet
- Einfügen von Texten, formatierten Elementen sowie ganzer Baumstrukturen möglich
- sofortige Darstellung der veränderten oder neuen Elemente

- Den Text in innerHTML erweitern , ganze HTML-Tags mit verwenden, damit gezeugt wird, dass das hier die HTMK-Seite in der HTML-Seite ist.
- Im Vergleich mit Einzeloperationen den verkürzten Code darstellen
- Siehe Übungsblatt 6 aus 200809! Aufgabe innerHTML



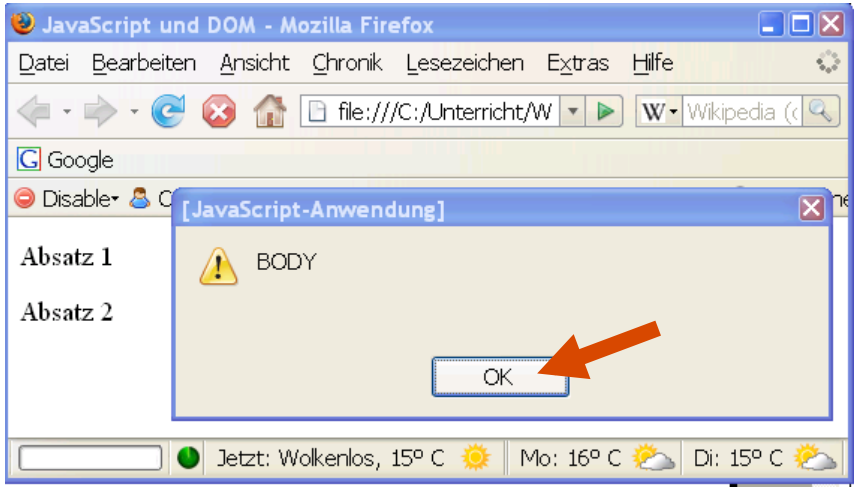
Eigenschaft : innerHTML

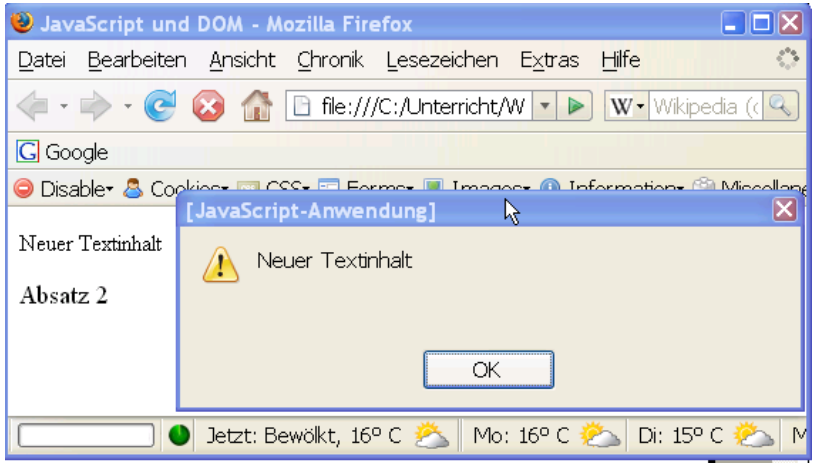
Beispiel:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de" lang="de">
<head>
         <title>JavaScript und DOM</title>
</head>
<body>
<p> <big> Absatz 1 </big></p>
<p> <big> Absatz 2 </big></p>
<script type="text/javascript">
alert(document.documentElement.childNodes[1].nodeName);
var e=document.body.childNodes[1];
e.innerHTML="Neuer Textinhalt";
alert(e.innerHTML);
</script>
</body>
</html>
```

Frage: Welche Ausgabe wird von dem Beispielprogramm erzeugt?

Eigenschaft : innerHTML





Antwort:

Dem ersten Elementknoten eines Dokumentes wurde ein neuer Inhalt zugewiesen.
Die Darstellung erfolgt sofort.
Der neue Inhalt wird auch in einem Alertfenster ausgegeben.

Vorsicht: Zugriff per Index auf die Elementknoten ist in unterschiedlichen Browsern unterschiedlich realisiert: beginnend bei 0 (Standard) oder bei 1; MSIE hält Standard ein, MF z.T. nicht)

Clientseitige Implementierungs-technologien (II):DOM

1. Einführung
2. Der Dokumentenbaum
3. Eigenschaften von Knoten
4. Methoden
5. Zugriff auf Elemente
    1. Eigenschaften
    2. **Elemente selektieren**
    3. Attribute bearbeiten
    4. Elemente erzeugen
6. Zusammenfassung

Elemente selektieren

Der Zugriff auf ein Element des `document`- Objektes kann über die CSS-ID, den Namen des Elementtyps oder über das `name`- Attribut erfolgen.

Methode	Beschreibung
`getElementById`	selektiert ein Element anhand einer CSS-ID
`getElementsByName`	selektiert alle Elemente als Array, die als `name`-Attribute das Argument der Methode enthalten
`getElementsByTagName`	selektiert alle Elemente als Array, die als Tagname das Argument der Methode enthalten

Methoden des DOM zur gezielten Selektion von Elementen

Elemente selektieren

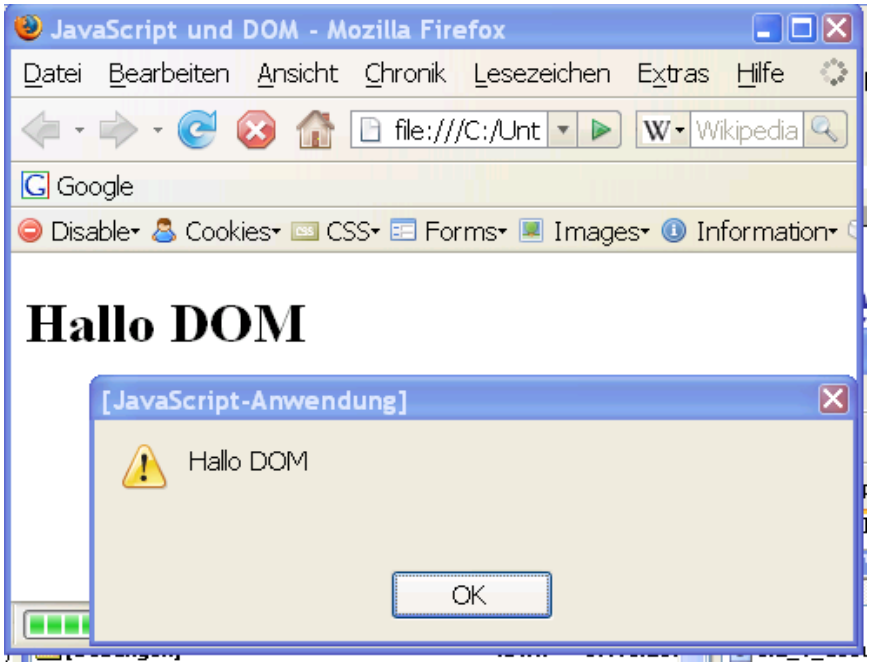
Beispiel zu getElementbyId

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de" lang="de">
<head>
   <title>JavaScript und DOM</title>
<script type="text/javascript">
window.onload = function()
{
    alert(
        document.getElementById("e1").firstChild.nodeValue
    );
}
</script>
</head>
<body>
<h1 id="e1">Hallo DOM</h1>
</body>
</html>
```

Frage 1: Welche Ausgabe wird von dem Beispielprogramm erzeugt?

Elemente selektieren

Antwort 1: *Der Textinhalt eines h1-Tags wird in einer Dialogbox angezeigt.*



Frage 2: *Mit welcher Befehlssequenz könnten Sie den Inhalt des selektierten Elements ändern? Schreiben Sie eine neue JavaScriptfunktion, welche die Ausgabe auf der Webseite von „Hallo DOM“ in „Neuer Textinhalt“ ändert!*

Elemente selektieren: Frage 2

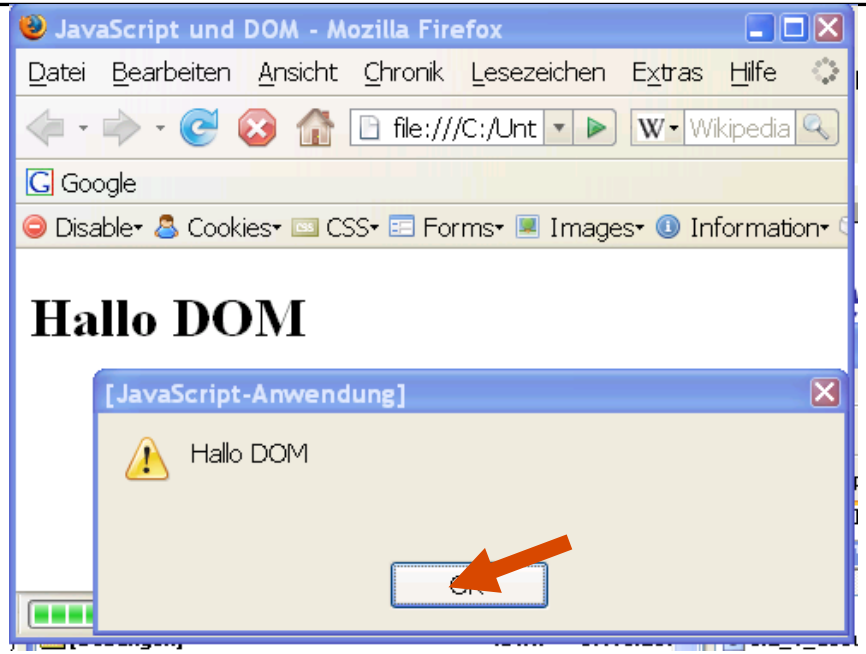
Frage 2: *Mit welcher Befehlssequenz könnten Sie den Inhalt des selektierten Elements ändern? Schreiben Sie eine neue JavaScriptfunktion, welche die Ausgabe auf der Webseite von „Hallo DOM“ in „Neuer Textinhalt“ ändert!*

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de" lang="de">
<head>
   <title>JavaScript und DOM</title>
<script type="text/javascript">
window.onload = function()
{
    alert(
        document.getElementById("e1").firstChild.nodeValue
    );
}
</script>
</head>
<body>
<h1 id="e1">Hallo DOM</h1>
</body>
</html>
```

Elemente selektieren

Lösung:

```
window.onload = function()
{
    alert(document.getElementById("e1").firstChild.nodeValue);
        var e = document.getElementById("e1");
        e.firstChild.nodeValue="Neuer Textinhalt";
}
```



Elemente selektieren

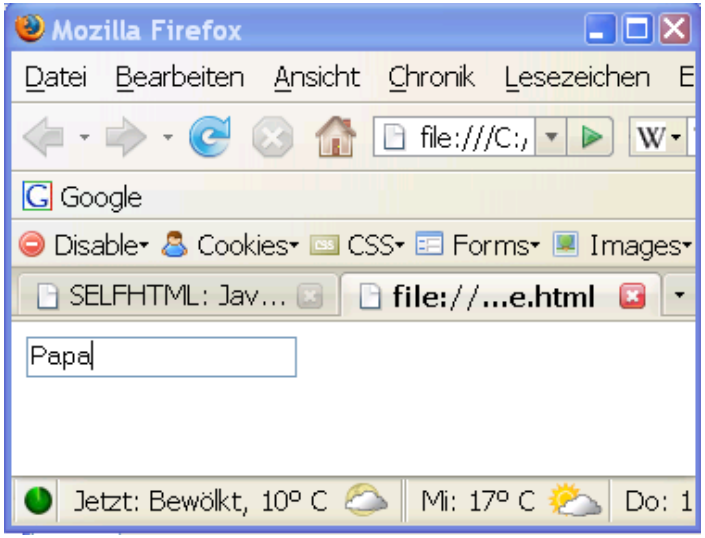
getElementsbyName

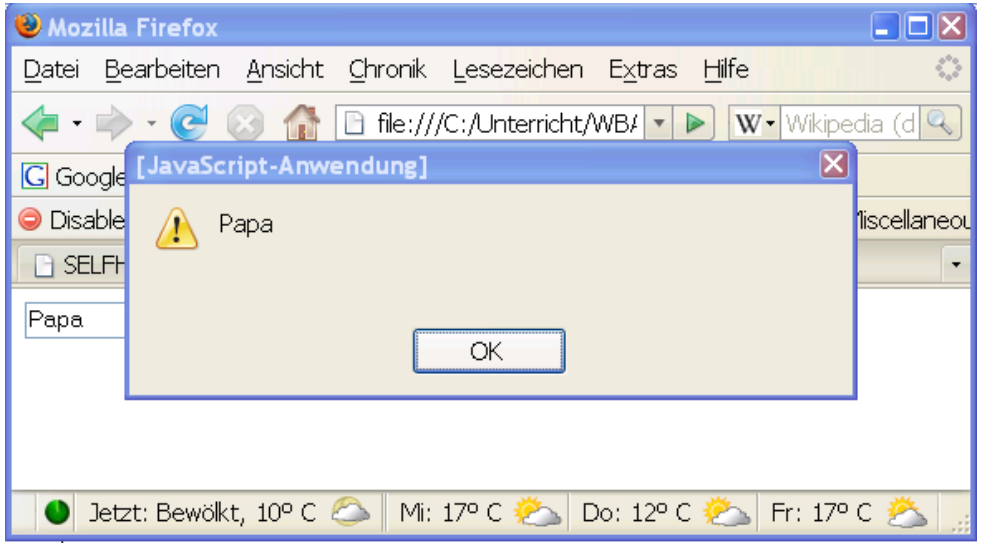
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de" lang="de">
<body>
<form>
<input type="text" name="inhalt“ onchange =
"alert(document.getElementsByName('inhalt')[0].value);" />
</form>
</body>
</html>
```

***Frage:* Welche Ausgabe wird von dem Beispielprogramm erzeugt?**

Elemente selektieren

getElementsbyName





Texteingabe und Enter

Antwort: *Bei verändertem Inhalt des Texteingabefeldes (Event:* `onchange`*) wird dieser Eingabetext im Meldefenster ausgegeben.*

Clientseitige Implementierungs-technologien (II):DOM

1. Einführung
2. Der Dokumentenbaum
3. Eigenschaften von Knoten
4. Methoden
5. Zugriff auf Elemente
    1. Eigenschaften
    2. Elemente selektieren
    3. **Attribute bearbeiten**
    4. Elemente erzeugen
6. Zusammenfassung

Attribute bearbeiten

Die Tags des XHTML besitzen Attribute, welche über die Methoden des DOM ausgelesen, editiert oder gelöscht werden können. Zusätzlich können neue Attribute dynamisch eingefügt werden.

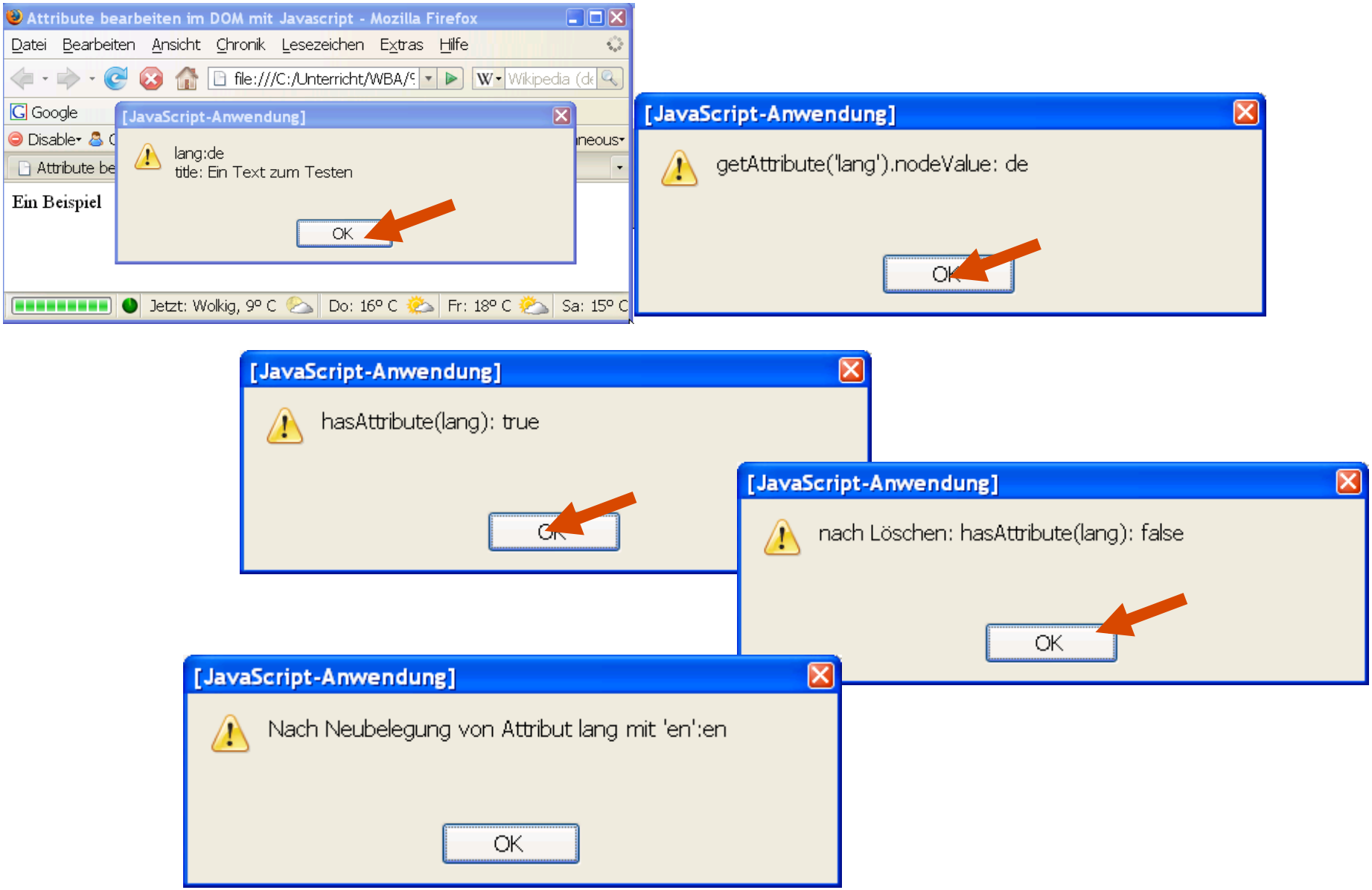
Methode	Argument	Beschreibung der Methode / des Arguments
`getAttribute`	Attribut	liefert Wert bzw. Inhalt eines Attributes / Attribut, nach dem gesucht werden soll
`getAttributeNode`	Attribut	liefert Objektreferenz auf den Attributknoten / Attribut, nach dem gesucht werden soll
`hasAttribute`	Attribute	gibt Booleschen Wert `true` oder `false` / Attribut, nach dem gesucht werden soll
`removeAttribute`	Attribute	löscht einen Attributeknoten aus einem Element / Attribut, welches gelöscht werden soll
`removeAttributeNode`	Objektreferenz	löscht Attributeknoten aus einem Element / Objektreferenz (z.B. von Methode `getAttributeNode`)
`setAttribute`	Attribute, Wert	fügt neues Attribute mit gewünschtem Wert ein / Name und Wert des neuen Attributes
`setAttributeNode`	Attribute	fügt einen neuen AttributeKnoten in ein Element ein / Name des neuen Attributes

Attribute bearbeiten: Beispiel

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de" lang="de">
<head><title>Attribute bearbeiten im DOM mit Javascript</title>
<script type="text/javascript">
window.onload = function()
{
      var e = document.getElementById("bsp");
      alert ("lang:"+e.getAttribute("lang")+"\ntitle: "+e.getAttribute("title"));
       alert("getAttribute('lang').nodeValue: "+e.getAttributeNode("lang").nodeValue);
       alert("hasAttribute(lang): "+e.hasAttribute("lang"));
       e.removeAttribute('lang');
       alert("nach Löschen: hasAttribute(lang): "+e.hasAttribute("lang"));
       e.setAttribute("lang","en");
       alert ("Nach Neubelegung von Attribut lang mit 'en':"+e.getAttribute("lang"));
}
</script>
</head>
<body>
<div id="bsp" title="Ein Text zum Testen" lang="de">
<big>Ein Beispiel</big>
</div>
</body<>
</html>
```

***Frage:* Beschreiben Sie die Ausgaben des Programms!**

Attribute bearbeiten: Beispiel



Clientseitige Implementierungs-technologien (II):DOM

1. Einführung
2. Der Dokumentenbaum
3. Eigenschaften von Knoten
4. Methoden
5. Zugriff auf Elemente
    1. Eigenschaften
    2. Elemente selektieren
    3. Attribute bearbeiten
    4. **Elemente erzeugen**
6. Zusammenfassung

Elemente erzeugen

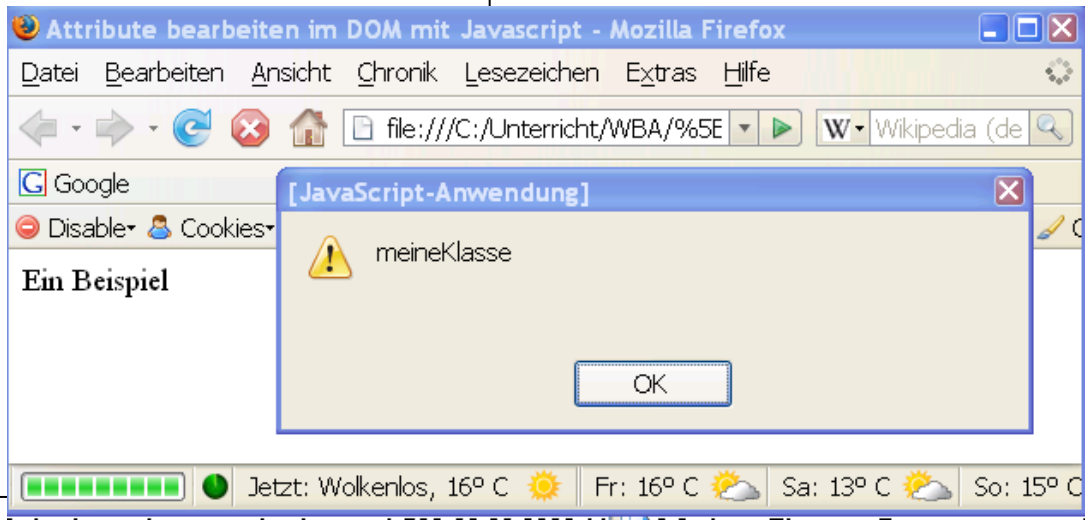
Ein wesentliches Merkmal des DOM ist es, neue Elemente dynamisch zu erzeugen und an eine beliebige Position im Dokumentenbaum einhängen zu können.

Methoden des DOM zur Erzeugung neuer Elemente:

Methode	Beschreibung der Methode
`createAttribute`	erzeugt für ein Element neuen Attributknoten, Attributname wird als Argument übergeben
`createElement`	erzeugt einen neuen xhtml-Elementknoten, der Name des gewünschten Tags wird als Argument übergeben
`createTextNode`	erzeugt einen neuen Textknoten und fügt gewünschten Text ein, der als Argument übergeben werden muss.

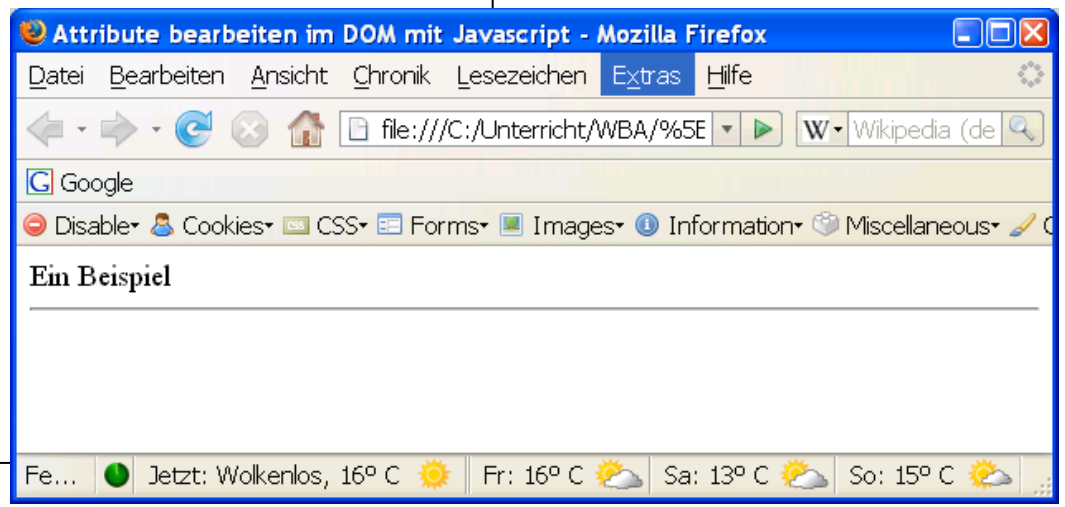
Elemente erzeugen: `createAttribute`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de" lang="de">
<head>
<title>Attribute bearbeiten im DOM mit Javascript</title>
<script type="text/javascript">
//<![CDATA[
window.onload = function()
{
      var e = document.getElementById("bsp");
      var a = document.createAttribute("class");
       a.nodeValue = "meineKlasse";
       e.setAttributeNode(a);
       alert(e.getAttribute("class"));
}
//]]>
</script>
</head>
<body>
<div id="bsp" title="Ein Text zum Testen"
lang="de">
<big>Ein Beispiel</big>
</div>
</body>
</html>
```



Elemente erzeugen: `createElement`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de" lang="de">
<head>
<title>Attribute bearbeiten im DOM mit Javascript</title>
<script type="text/javascript">
//<![CDATA[
window.onload = function()
{
      var e = document.getElementById("bsp");
      var hr = document.createElement("hr");
      e.appendChild(hr);
}
//]]>
</script>
</head>
<body>
<div id="bsp" title="Ein Text zum Testen"
  lang="de">
<big>Ein Beispiel</big>
</div>
</body>
</html>
```

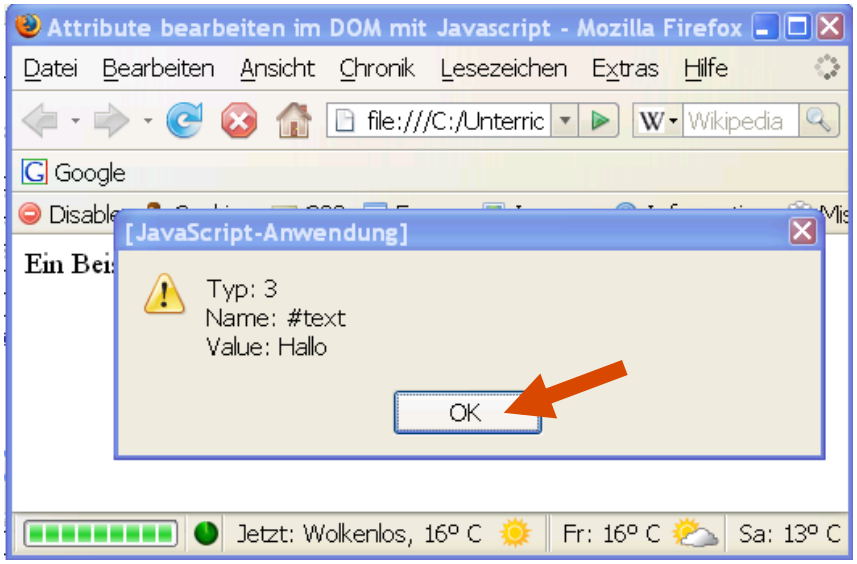


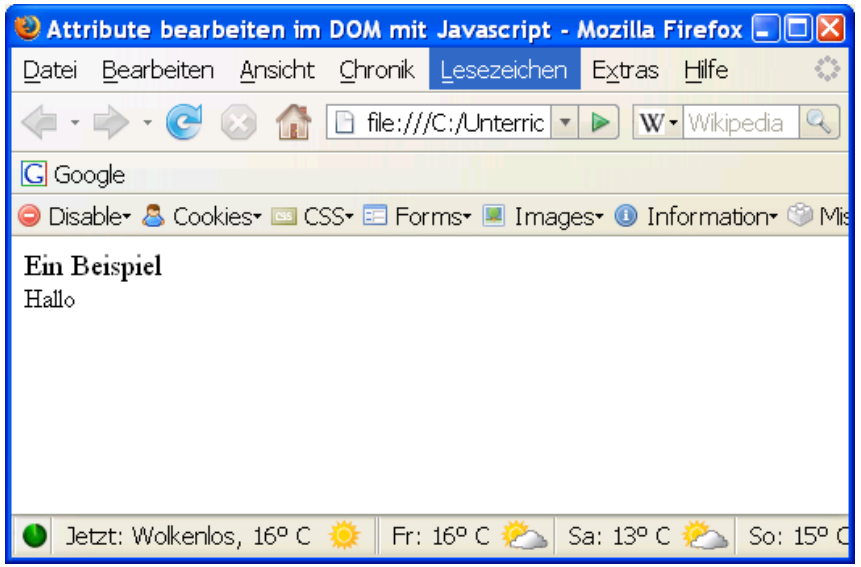
Elemente erzeugen: createTextnode

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de" lang="de">
<head>
<script type="text/javascript">
//<![CDATA[
window.onload = function()
{
   var txt= document.createTextNode("Hallo");
   alert("Typ: " +txt.nodeType+"\nName: " +txt.nodeName+"\nValue: " +txt.nodeValue)
   document.body.appendChild(txt);
}
//]]>
</script>
</head>
<body>
<div id="bsp" title="Ein Text zum Testen" lang="de">
<big>Ein Beispiel</big>
</div>
</body>
</html>
```

Elemente erzeugen:

createTextnode





Clientseitige Implementierungs-technologien (II):DOM

1. Einführung
2. Der Dokumentenbaum
3. Eigenschaften von Knoten
4. Methoden
5. Zugriff auf Elemente
    1. Eigenschaften
    2. Elemente selektieren
    3. Attribute bearbeiten
    4. Elemente erzeugen
6. **Zusammenfassung**

Zusammenfassung

- Dokumente werden im DOM-Modell in einer Baumstruktur dargestellt. Die Knoten stehen über Verwandtschaftsbeziehungen zueinander in Verbindung.
- Jeder Elementknoten entspricht genau einem XML- oder HTML-Element
- Ein Attributknoten entspricht genau einem Attribut in XML oder HTML; sie sind Eigenschaften von Elementknoten
- Ein Textknoten stellt den textuellen Inhalt eines Elements oder Attributs dar
- DOM erlaubt :
  - die Navigation zwischen den Elementknoten
  - das Erzeugen, Verschieben und Löschen von Knoten
  - Das Auslesen, Ändern und Löschen von Textinhalten

Literatur für DOM

1. http://developer.mozilla.org/en/docs/DOM
2. http://developer.mozilla.org/en/doc/JavaScript
3. http://www.w3.org/XML/
4. Johannes Gamperl: „Ajax Grundlagen“, Herdt- Verlag Bildungsmedien GmbH, Bodenheim 2006
5. *http://de.wikipedia.org/wiki/Document_Object_Model*

Ausblick:

1. Einführung in WBA
2. Wiederholung: Grundlagen des WWW, HTML und HTTP
3. **Clientseitige Implementierungstechnologien:** Javascript, DOM, **Ajax**, Jquery, (Java-Applet)