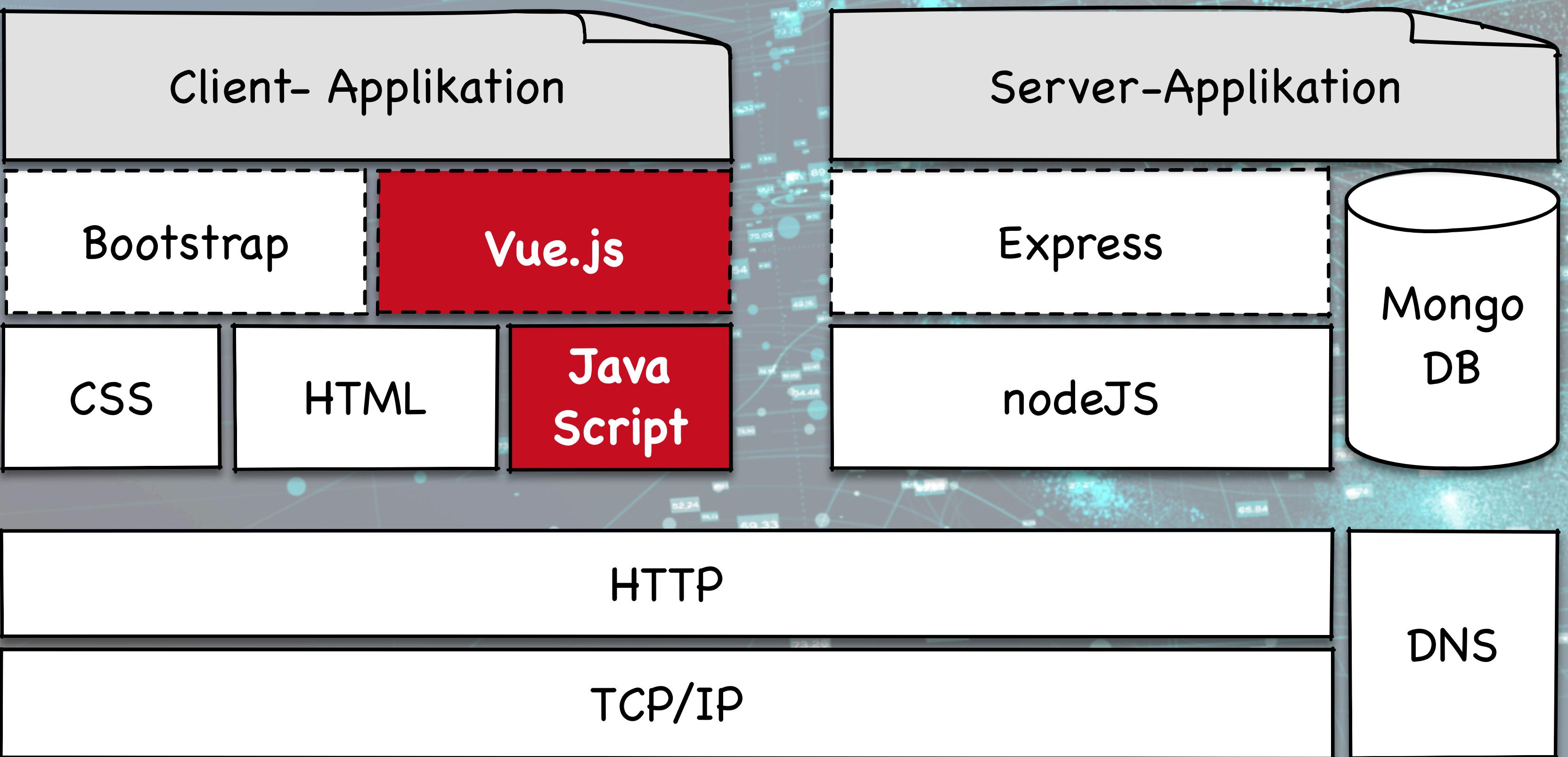


WEB TECHNOLOGIEN 2022

KAPITEL 5: JAVASCRIPT IM BROWSER

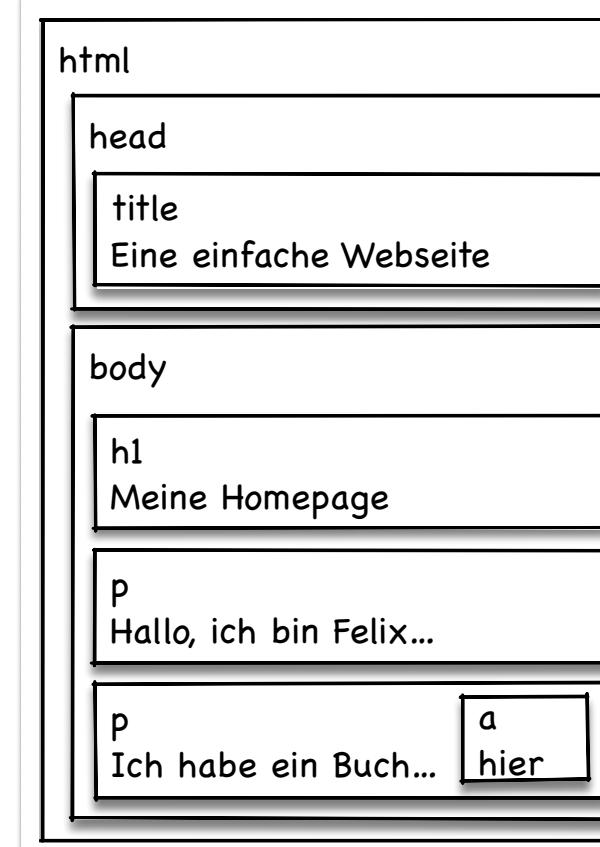
PROF. DR. AXEL KÜPPER
FACHGEBIET SERVICE-CENTRIC NETWORKING I TU BERLIN & T-LABS

WEBTECHNOLOGIEN ÜBERBLICK



5.1 DAS DOCUMENT OBJECT MODEL ÜBERBLICK

```
1 <html>
2   <head>
3     <title>Eine einfache Webseite</title>
4   </head>
5   <body>
6     <h1>Meine Homepage</h1>
7     <p>Hallo, ich bin Felix und dies ist
8       meine Homepage.</p>
9     <p>Ich habe ein Buch geschrieben. Lies es
10    <a href="http://buch.de">hier</a>.</p>
11   </body>
12 </html>
```



"USE CASES" DES DOM

- Navigation zwischen den einzelnen Knoten
- Erzeugen, Verschieben und Löschen von Knoten
- Auslesen, Ändern und Löschen von Inhalten
- Veränderungen der CSS-Eigenschaften von Elementen
- Definition, Auslösen und Abfangen von Ereignissen (Events)

DOCUMENT OBJECT MODEL

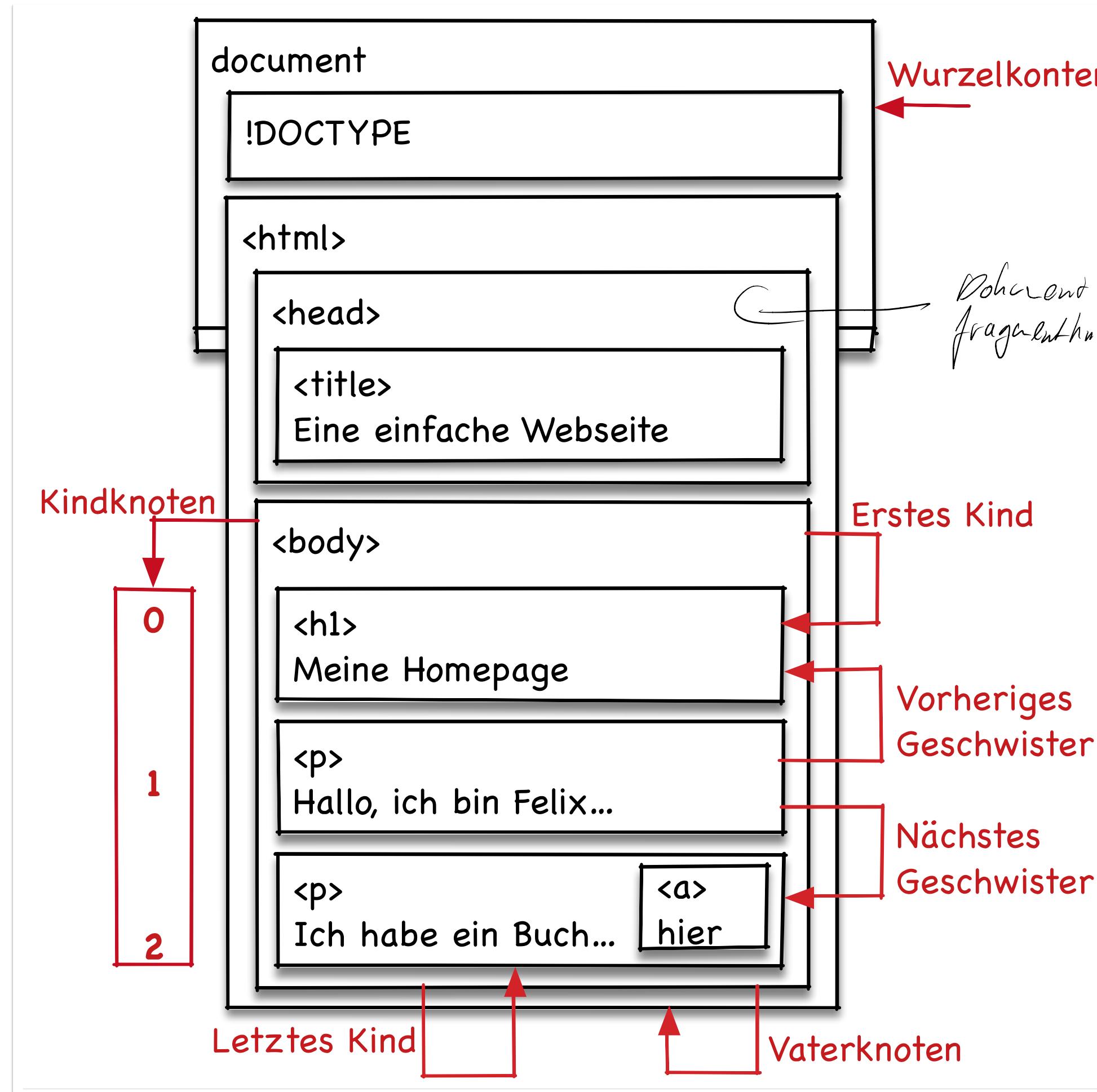
- Spezifikation von Schnittstellen (*Application Programming Interface, API*) für den Zugriff auf HTML- (oder XML-) Dokumente
- Implementierung besteht aus einem Satz von Objekten zusammen mit deren Methoden und Eigenschaften
- JavaScript-Programme können dynamisch den Inhalt, die Struktur und das Layout des Dokuments ändern

VERARBEITUNG EINES DOKUMENTS

- Bestehendes Dokument wird durch den Browser im Speicher als DOM abgebildet und steht für JavaScript-Programme als `document`-Objekt zur Verfügung
- Anhand `document` kann durch Methoden und Eigenschaften der API auf den DOM zugegriffen werden

5.1 DAS DOCUMENT OBJECT MODEL

BESTANDTEILE DES DOM



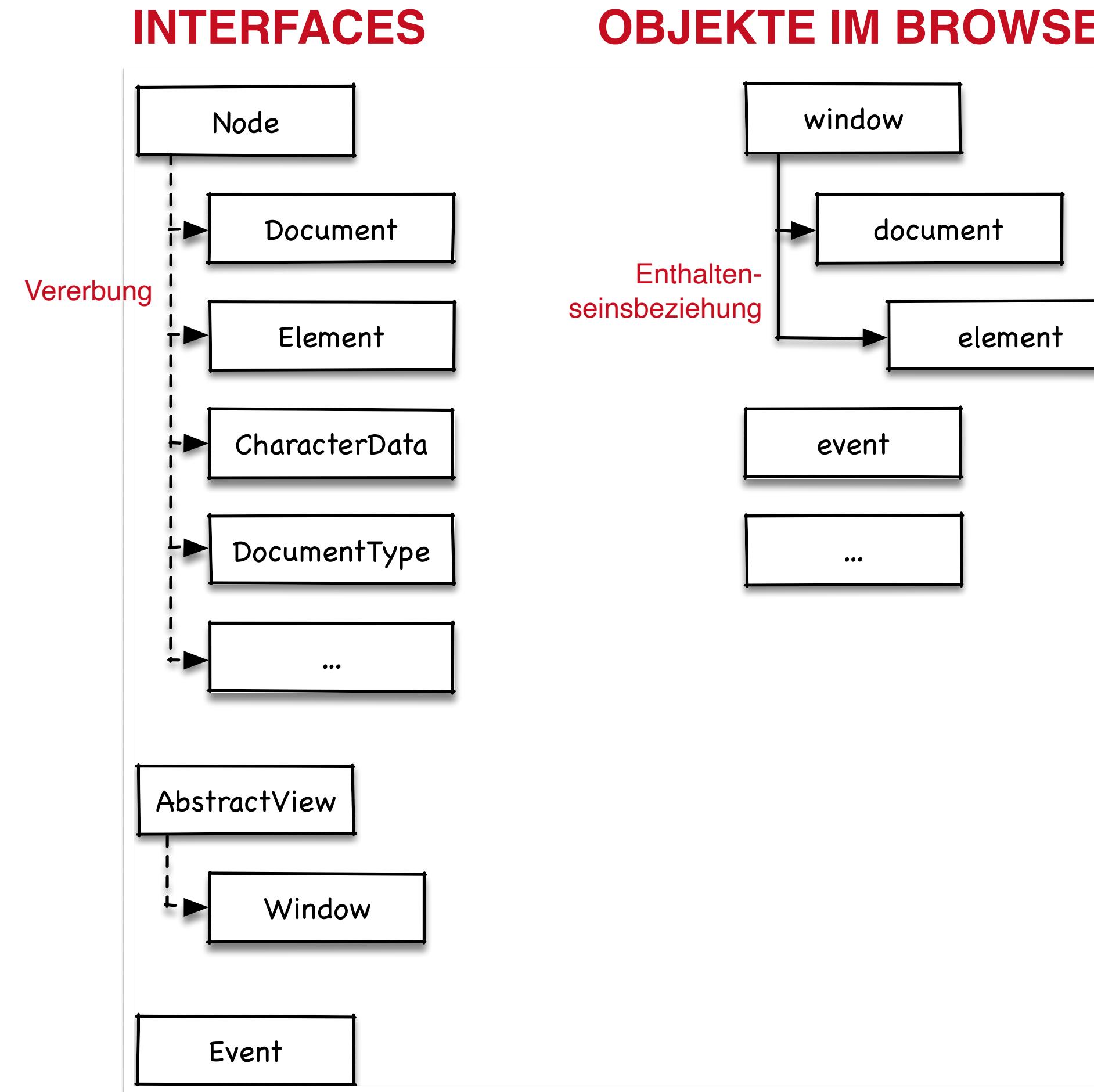
ARTEN VON KNOTEN

- Dokumentenknoten, repräsentiert durch das Objekt `document`, stellt das gesamte HTML-Dokument dar
- Dokumentfragmentknoten stellt einen Teil der Baumstruktur dar
- Elementknoten, repräsentiert durch das Objekt `element`, entspricht exakt einem HTML-Element
- Attributknoten entspricht einem Attribut in einem HTML-Element
- Textknoten stellt den textuellen Inhalt eines Elements dar

ARTEN VON BEZIEHUNGEN

- Wurzelknoten an oberster Stelle der Hierarchie, repräsentiert durch `document`
- Kindknoten als Nachfolger eines Knoten, durchnummierter
- Vaterknoten als Vorgänger eines Knoten
- Vorheriges Geschwister, d.h. Nachfolge-Nachbarknoten
- Nachfolgendes Geschwister, d.h. Vorgänger-Nachbarknoten

5.1 DAS DOCUMENT OBJECT MODEL DIE DOM-SPEZIFIKATION

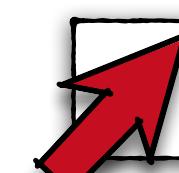


INTERFACES

- W3C-DOM definiert Datentypen als Schnittstelle (Interface) für den Zugang zum DOM
- Interfaces definieren Methoden und Eigenschaften und repräsentieren den Prototyp der zu instanzierenden Objekte
- Schreibweise: Interfaces werden groß, Objekte klein geschrieben
- Browser-Laufzeitumgebung stellt window und document (und andere Objekte) für den DOM-Zugriff bereit

ZWEI ARTEN VON BEZIEHUNGEN

- Vererbung (Inheritance): Vererbung von Methoden und Eigenschaften an das erbende Interface
- Enthaltenseinsbeziehung: Schachtelung von Objekten



https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model

5.1 DAS DOCUMENT OBJECT MODEL EINBINDUNG VON JAVASCRIPT (I)

```
2 <head>
3   <title>JavaScript-Test</title>
4   <script>
5     function meineFunktion() {
6       document.getElementById("meinTest").innerHTML =
7         "Ein JavaScript wurde ausgeführt!";
8     }
9   </script>
10  </head>
11  <body>
12    <h1>Ein JavaScript bei der Ausführung</h1>
13    <p id="meinTest">JavaScript testen</p>
14    <button type="button" onclick="meineFunktion()">
15      JavaScript starten
16    </button>
17  </body>
```

```
2 <head>
3   <title>JavaScript-Test</title>
4   </head>
5   <body>
6     <h1>Ein JavaScript bei der Ausführung</h1>
7     <p id="meinTest">JavaScript testen</p>
8     <button type="button" onclick="meineFunktion()">
9       JavaScript starten
10    </button>
11    <script>
12      function meineFunktion() {
13        document.getElementById("meinTest").innerHTML =
14          "Ein JavaScript wurde ausgeführt!";
15      }
16    </script>
17  </body>
```

IM <head> VON HTML

- JavaScript-Programm befindet sich im Header des HTML-Dokuments innerhalb des <script>-Elements

IM <body> VON HTML

- JavaScript-Programm befindet sich im Body des HTML-Dokuments innerhalb des <script>-Elements
- Anweisungen auf globaler Ebene werden sofort nach Aufbau der HTML-Seite im Browser ausgeführt
- Funktionen werden nur bei Aufruf ausgeführt

5.1 DAS DOCUMENT OBJECT MODEL EINBINDUNG VON JAVASCRIPT (II)

```
1 <html>
2   <head>
3     <title>JavaScript-Test</title>
4   </head>
5   <body>
6     <h1>Ein JavaScript bei der Ausführung</h1>
7     <p id="meinTest">JavaScript testen</p>
8     <button type="button"
9        onclick="document.getElementById('meinTest').innerHTML=
10          'Ein JavaScript wurde ausgeführt!'">
11       JavaScript starten
12     </button>
13   </body>
14 </html>
```

INNERHALB EINES HTML-STARTTAGS

- JavaScript-Programm ist Wert eines HTML-Attributs
- Umständlich und schwer lesbar

```
1 <html>
2   <head>
3     <title>JavaScript-Test</title>
4     <script src="js/meinScript.js"></script>
5   </head>
6   <body>
7     <h1>Ein JavaScript bei der Ausführung</h1>
8     <p id="meinTest">JavaScript testen</p>
9     <button type="button" onclick="meineFunktion()">
10       JavaScript starten
11     </button>
12   </body>
13 </html>
```

```
1 function meineFunktion() {
2   document.getElementById("meinTest").innerHTML =
3   "Ein JavaScript wurde ausgeführt!";
4 };
```

ALS EXTERNE DATEI

- Einbindung des JavaScript-Programms als externe Datei im Header oder im Body
- Eine einzelne JavaScript-Datei für alle Webseiten einer Website

5.1 DAS DOCUMENT OBJECT MODEL

EIN ERSTES BEISPIEL

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>DOM Schnittstellen</title>
5      </head>
6      <body>
7          <h1>Die DOM Schnittstelle</h1>
8          <p id="msg">Der Absatztext</p>
9
10         <script>
11             var text = document.getElementById("msg").innerHTML;
12             text += " " + "wurde erweitert!";
13             document.getElementById("msg").innerHTML = text;
14         </script>
15
16     </body>
17 </html>
```

/chapter05/example01/index01.html



ELEMENTE FINDEN

- Verschiedene Methoden und Eigenschaften für den Zugriff auf die Knoten des DOM-Baumes
- Methode getElementById() ermöglicht Zugriff auf HTML-Elemente deren id-Attribut einen bestimmten Wert enthält
- Eigenschaft innerHTML ermöglicht das Auslesen oder Ersetzen des Inhalts eines HTML-Elements

5.1 DAS DOCUMENT OBJECT MODEL

AUFGINDEN VON ELEMENTEN IM DOCUMENT-OBJEKT

AUFGIND-METHODEN IN document

Methoden in document	Beschreibung
document.getElementById()	Liefert das Element dessen id-Attribut den gewünschten Wert hat.
document.getElementsByClassName()	Liefert eine Liste von Elementen dessen class-Attribute den spezifizierten Namen haben.
document.getElementsByTagName()	Liefert eine Liste von Elementen deren Tag den spezifizierten Namen haben. Der Universalselektor * liefert Referenzen auf alle Elemente des Dokuments.
document.getElementsByName()	Liefert eine Liste mit Elementen mit einem bestimmten name-Attribut
document.querySelector(s)	Liefert das erste Element (und nur dieses), das dem angegebenen CSS-Selektor s entspricht
document.querySelectorAll(s)	Liefert eine Liste mit allen Elementen zurück, die dem angegebenen CSS-Selektor entsprechen

AUFGIND-EIGENSCHAFTEN IN document

Eigenschaften in document	Beschreibung
document.anchors	Liefert eine Liste aller Anchor-Elemente (gekennzeichnet durch <a>-Tags).
document.body	Liefert das body-Element des Dokumentes.
document.doctype	Liefert den Wert von <!DOCTYPE> des Dokuments.
document.documentElement	Liefert das html-Element
document.forms	Liefert eine Liste aller Formulare (gekennzeichnet durch <form>-Tags) des Dokuments.
document.images	Liefert eine Liste aller Bilder (gekennzeichnet durch -Tag) des Dokuments.
document.links	Liefert eine Liste mit allen Verweisen die in dem Dokument enthalten sind (gekennzeichnet durch <a>, <area> und andere Tags).

- Hinweis: Methoden in diesen Tabellen geben ein Objekt nodeList zurück, kein Array

5.1 DAS DOCUMENT OBJECT MODEL

AUFFINDEN VON ELEMENTEN IM ELEMENT-OBJEKT

AUFFIND-METHODEN/EIGENSCHAFTEN IN element

Eigenschaften/Methoden	Beschreibung
element.attributes	Liefert eine Liste mit allen Attributen eines Elements.
element.childNodes	Liefert eine Liste mit allen Kindelementen eines Elements.
element.firstChild	Liefert das erste Kind eines Elements.
element.getAttribute()	Liefert den Wert des spezifizierten Attributs.
element.getElementsByTagName()	Liefert eine Liste mit allen Elementen mit dem Spezifizierten Tag-Namen.
element.hasAttribute()	Liefert true wenn ein Element das spezifizierte Attribut hat, sonst false.
element.hasAttributes()	Liefert true wenn ein Element (beliebige) Attribute hat, sonst false.
element.hasChildNodes()	Liefert true falls das Element Kindelemente hat, sonst false.
element.lastChild	Liefert das letzte Kindelement eines Elements.
element.nextSibling	Liefert das Geschwister welches dem Element unmittelbar folgt.

Eigenschaften/Methoden	Beschreibung
element.ownerDocument	Liefert das Wurzel-Element des Elements (üblicherweise document).
element.parentNode	Liefert den Elternknoten des Elements.
element.previousSibling	Liefert das Geschwister welches dem Element unmittelbar vorangeht.

5.1 DAS DOCUMENT OBJECT MODEL

AUFFINDEN VON ELEMENTEN



```
4 <body>
5   <article id="lead">
6     <h1>Die DOM Schnittstelle</h1>
7     <p>Erster Absatztext im Artikel</p>
8     <p>Zweiter Absatztext im Artikel</p>
9   </article>
10  <p>Erster Absatztext außerhalb des Artikels</p>
11  <p>Zweiter Absatztext außerhalb des Artikels</p>
12  <p id="result"></p>
13  <script>
14    var text = "";
15    var p_elements = document.getElementsByTagName("p");
16    var text = "Insgesamt sind " + p_elements.length + " p-Elemente im Dokument";
17    var article_elements = document.getElementById("lead");
18    if(article_elements) {
19      var art_p_elements = article_elements.getElementsByTagName("p");
20      text += " und " + art_p_elements.length + " im article-Element enthalten!";
21    }
22    else {
23      text += " enthalten!";
24    }
25    text += " Der zweite Absatz im Artikel lautet: " + art_p_elements[1].innerHTML;
26    document.getElementById("result").innerHTML = text;
27  </script>
28 </body>
```

5.1 DAS DOCUMENT OBJECT MODEL MANIPULATION VON ELEMENTEN

```
4 <body>
5   <h1 id="headline">Überschrift</h1>
6   <p id="mainarticle">Absatztext</p>
7   <button onclick="changeContent()">Ändern mit innerHTML</button>
8
9   <script>
10    function changeContent() {
11      document.getElementById("headline").innerHTML = "Neue Überschrift!";
12      var elem = document.getElementById("mainarticle");
13      elem.innerHTML = "Neuer Inhalt für den Absatztext";
14    }
15  </script>
16 </body>
```



METHODEN UND EIGENSCHAFTEN ZUM VERÄNDERN VON ELEMENTEN

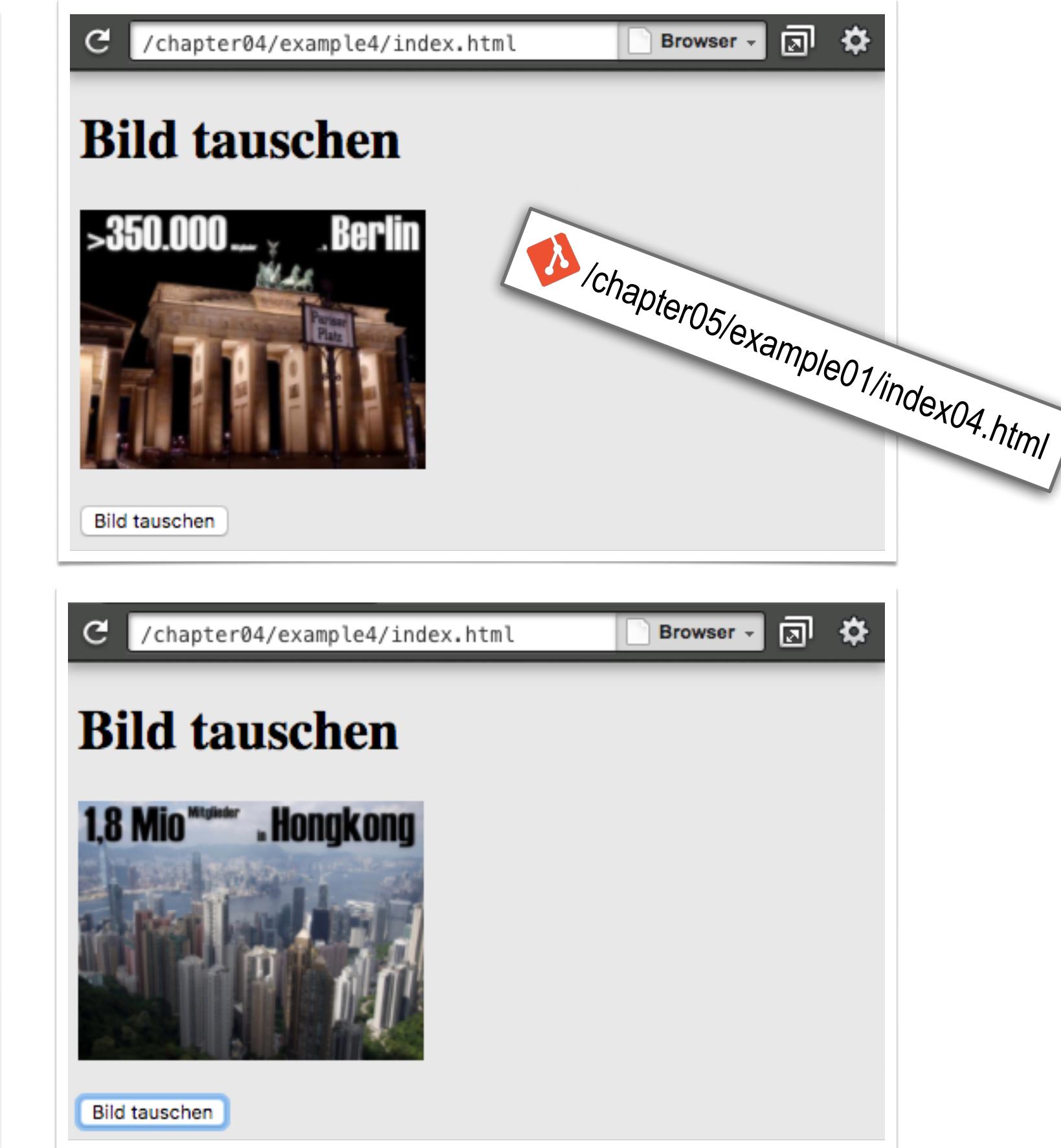
Eigenschaften/Methoden	Beschreibung
document.write	Schreibt Inhalt in ein HTML-Dokument. Achtung: Bestehender Inhalt wird überschrieben.
element.innerHTML	Ändert den Inhalt eines HTML-Elements.
element.getAttribute	Ändert den Wert eines HTML-Attributs.
element.setAttribute()	Ändert ebenfalls den Wert eines HTML-Attributs.
element.style.property	Ändert den CSS-Style eines HTML-Elements.
element.appendChild()	Fügt einen neuen Kindknoten an letzter Stelle der Liste der Kindknoten zu einem Element hinzu.
element.insertBefore()	Fügt einen neuen Kindknoten vor dem spezifizierten Kindknoten ein.
element.removeAttribute()	Entfernet ein Attribut aus dem spezifizierten Element.
element.removeChild()	Entfernt den spezifizierten Kindknoten.
element.replaceChild()	Ersetzt den spezifizierten Kindknoten durch einen neuen Kindknoten.
element.textContent	Setzt oder liefert den textuellen Inhalt des Elements und aller direkten und indirekten Nachfolger.

<https://developer.mozilla.org/en-US/docs/Web/API/Document>

<https://developer.mozilla.org/en-US/docs/Web/API/Element>

5.1 DAS DOCUMENT OBJECT MODEL MANIPULATION VON ATTRIBUTWERTEN

```
4 <body>
5   <h1>Bild tauschen</h1>
6   <p></p>
7
8   <button onclick="changePicture()">Bild tauschen</button>
9
10  <script>
11    var xchange = true;
12    function changePicture() {
13      var current = document.getElementById("pic");
14
15      if(xchange) {
16        current.src = "img/img2.jpg";
17        current.alt = "Bild02";
18        xchange = false;
19      }
20      else {
21        current.src = "img/img1.jpg";
22        current.alt = "Bild01";
23        xchange = true;
24      }
25    }
26  </script>
27 </body>
```



5.1 DAS DOCUMENT OBJECT MODEL

MANIPULATION VON STYLES

```
4 <body>
5   <h1 id="headline">HTML Style ändern</h1>
6   <p id="mainarticle">Ein einfacher Absatztext ...</p>
7   <button onclick="changeColor()">Farbe ändern</button>
8
9   <script>
10    var element = document.getElementById("mainarticle");
11    element.style.color = "white";
12    element.style.background = "black";
13
14    function changeColor() {
15      document.getElementById("headline").style.color = "gray";
16      document.getElementById("headline").style.fontStyle = "italic";
17    }
18  </script>
19 </body>
```



style-OBJEKT

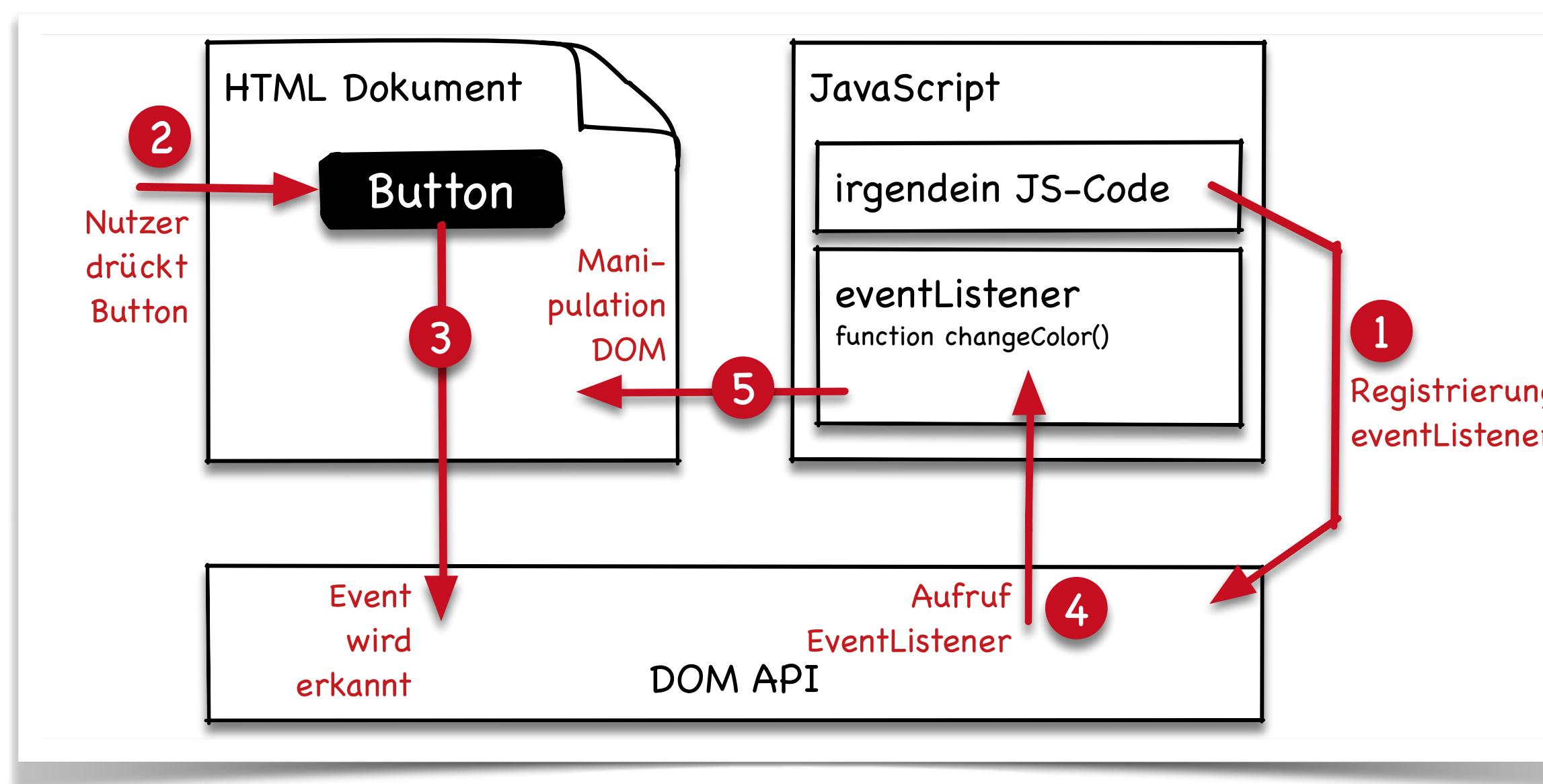
- Änderung der CSS-Eigenschaften von Elementen in JavaScript
- Objekt enthält Eigenschaften, die denen von CSS entsprechen

CAMEL CASE

- JavaScript erlaubt nicht die Verwendung von Bindestrichen in Eigenschafts- und Variablenamen ("-" wird als Minus interpretiert)
- Camel-Case-Schreibweise für CSS-Eigenschaften in JavaScript
- Beispiele: fontStyle statt font-style und borderTopRightRadius statt border-top-right-radius



5.2 EREIGNISSE ÜBERBLICK



EREIGNISSE

- Interaktive Webseiten mit Hilfe von Ereignissen (Events)
- Beispiel: Wechsel von Bildern beim Überfahren mit der Maus
- Ereignisse werden durch die Laufzeitumgebung erkannt oder können durch das Programm ausgelöst werden
- Event Handler oder Event Listener sind Callbacks, die ausgeführt werden wenn ein bestimmtes Ereignis auf einem Element oder Objekt ausgelöst wurde

ABLAUF

- (1) Registrierung des Event Listener bei der Laufzeitumgebung für Ereignisse auf bestimmten Elementen bzw. Objekten
- (2) Auftreten des Ereignisses, zum Beispiel Betätigung eines Buttons mit der Maus
- (3) Erkennung des Ereignisses durch die Laufzeitumgebung
- (4) Benachrichtigung des Event Listener
- (5) Durchführung ereignisbezogener Aktionen

5.2 EREIGNISSE

EVENT LISTENER (I)

```
6  <!-- vorheriges HTML -->
7  <button onclick="changeColor()">Farbe ändern</button>
8  <!-- weiteres HTML -->
9
10 <script>
11 function changeColor() {
12     // Funktionslogik
13 }
14 </script>
```

```
1 var element = document.getElementById("eineID");
2 element.onmouseover=function() /*Funktionslogik*/;
```

oder

```
1 var element = document.getElementById("eineID");
2 element.onmouseover=changeColor;
3 function changeColor() {
4     //Funktionslogik
5 }
```

EVENT LISTENER ALS HTML-ATTRIBUT

- Einrichten des Event Listener durch Ereignis-Attribut des jeweiligen Elements
- Bei einem Ereignis auf dem Element wird hinterlegte Funktion aufgerufen
- Vermischung von JavaScript und HTML
- Nur auf HTML-Elemente anwendbar
- Jeweils nur ein Event Listener pro Ereignistyp und Element

EVENT LISTENER ALS EIGENSCHAFT EINES OBJEKTES

- Einrichten des Event Listener durch eine Ereignis-Eigenschaft im Objekt, welches das Element repräsentiert
- Jeweils nur ein Event Listener pro Ereignistyp und Element - bei Registrierung mehrerer Event Listener werden vorherige überschrieben

5.2 EREIGNISSE

EVENT LISTENER (II)

```
5  <!-- vorheriges HTML -->
6  <button id="but01">Farbe ändern</button>
7  <!-- weiteres HTML -->
8  <script>
9    var element = document.getElementById("but01");
10   element.addEventListener("click", changeColor);
11   element.addEventListener("click", changeText);
12
13   function changeColor() {
14     // Funktionslogik
15   }
16
17   function changeText() {
18     // Funktionslogik
19   }
20 </script>
```

EVENT LISTENER MIT addEventListener()

- Verknüpfung zwischen Element und Event Listener mittels der Methode `addEventListener()` auf dem jeweiligen Element
- Erster Parameter bezeichnet die Art des Events (ohne den Präfix "on")
- Zweiter Parameter enthält die Referenz des Event Listener
- Erster Parameter des Event Listener ist eine Referenz auf ein Objekt vom Typ Event, welches weitere Informationen über das Event liefert
- Beispiel: `currentTarget` von Event bezeichnet das Element auf das sich das Ereignis bezieht (siehe nächste Folie)

5.2 EREIGNISSE

PARAMETER DES EVENT LISTENER

```
5  <!-- vorheriges HTML -->
6  <button onclick="changeColor(event)">Farbe ändern</button>
7  <!-- weiteres HTML -->
8
9  <script>
10     function changeColor(evt) {
11         // Funktionslogik
12     }
13 </script>
```

```
1 var element = document.getElementById("eineID");
2 element.onmouseover=changeColor;
3 function changeColor(evt) {
4     // Funktionslogik
5 }
```

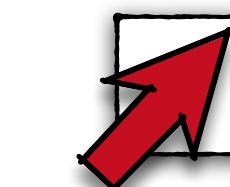
```
1 element.addEventListener("click", changeColor);
2 element.addEventListener("click", changeText);
3 function changeColor(evt) {
4     // Funktionslogik
5 }
6 function changeText(evt) {
7     // Funktionslogik
8 }
```

PARAMETER DES EVENT LISTENERS

- Erster Parameter des Event Listener referenziert ein event-Objekt, das beim Aufruf des Listener durch die Laufzeitumgebung übergeben wird
- Übergabeparameter muss nicht notwendigerweise im Funktionskopf deklariert werden
- event-Object: Methoden und Eigenschaften zur Abfrage von Daten, die das Event näher beschreiben

EIGENSCHAFTEN DES event-OBJEKTES

Eigenschaften	Beschreibung
event.currentTarget	Liefert das Element welches das Ereignis ausgelöst hat.
event.timeStamp	Liefert den Zeitstempel im ms zu dem das Ereignis auftrat.
event.clientX und event.clientY	Koordinaten des Mauszeigers zum Zeitpunkt eines aufgetretenen Maus-Ereignisses.
event.key	Liefert Wert der Taste die bei einem Tastatur-Ereignis ausgelöst wurde.



<https://developer.mozilla.org/en-US/docs/Web/API/Event>

5.2 EREIGNISSE

EREIGNISTYPEN (I)

FENSTEREREIGNISSE

Ereignis	Wird ausgelöst...
onafterprint	nachdem das Dokument ausgedruckt wurde.
anbeforeprint	bevor das Dokument ausgedruckt wird.
onbeforeunload	bevor das Dokument verlassen wird (d.h. kurz bevor der Browser geschlossen wird oder der Nutzer eine andere Webseite aufruft).
onerror	wenn ein Fehler auftritt.
onhaschange	wenn das Dokument verändert wurde.
onload	wenn die Webseite "fertig" geladen ist.
onmessage	wenn eine Nachricht ausgelöst wurde.
onoffline	wenn die Internetverbindung unterbrochen wurde.
online	wenn die Internetverbindung hergestellt wurde.
onpagehide	wenn das Fenster welches das Dokument anzeigt versteckt wird.
onpageshow	wenn das Fenster welches das Dokument anzeigt sichtbar wird.
onpopstate	wenn sich die Historie des Fensters ändert.
onredo	wenn das Dokument einen Vorgang "Nochmals tun" ausführt.
onresize	wenn die Größe des Browserfensters verändert wird.
onstorage	wenn eine gespeicherte Webseite aktualisiert wird.
onundo	wenn das Dokument einen Vorgang "Rückgängig machen" ausführt.
onunload	wenn das Dokument verlassen wurde oder das Browserfenster geschlossen wird.

TASTATUREREIGNISSE

Ereignis	Wird ausgelöst...
onkeydown	der Nutzer eine Taste gedrückt hält.
onkeypress	wenn der Nutzer eine Taste drückt.
onkeyup	wenn der Nutzer eine Taste loslässt.

MAUSEREIGNISSE

Ereignis	Wird ausgelöst...
onclick	bei einem Mausklick auf das Element.
ondblclick	bei einem Maus-Doppelklick auf das Element.
ondrag	wenn das Element mit der Maus gezogen wird.
ondragend	beim Beenden des Drag-Vorgangs.
ondragenter	wenn eine Element auf ein Zielelement gezogen wurde.
ondragleave	wenn ein Element das Zielelement verlässt.
ondragstart	beim Beginn eines Drag-Vorgangs
ondrop	wenn das gezogene Element fallen gelassen wird.
onmousedown	wenn die Maustaste auf einem Element gedrückt wird.
onmousemove	wenn der Mauszeiger sich über ein Element bewegt.
onmouseout	wenn der Mauszeiger sich aus einem Element heraus bewegt.
onmouseover	wenn sich der Mauszeiger über einem Element bewegt.
onmouseup	wenn die Maustaste losgelassen wird.
onmousewheel	wenn das Mausrad betätigt wird.
onscroll	wenn der Scrollbalken eines Elements bewegt wird.

5.2 EREIGNISSE

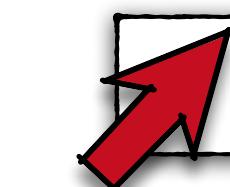
EREIGNISTYPEN (II)

FORMULAREREIGNISSE

Ereignis	Wird ausgelöst...
onblur	in dem Moment wenn eine Element den Fokus verliert.
onchange	in dem Moment wenn sich der Wert eines Elementes ändert.
oncontextmenu	wenn eine Kontextmenü ausgelöst wird (nicht unterstützt in den meisten Browsern).
onfocus	wenn eine Element den Fokus erhält.
onformchange	wenn sich ein Formular verändert.
onforminput	wenn das Formular mit Nutzerinhalten gefüllt wird.
oninput	wenn ein Element mit Nutzerinhalten gefüllt wird.
oninvalid	wenn ein Element ungültig ist.
onselect	nachdem Text in einem Element ausgewählt wurde.
onsubmit	ein Formular abgeschickt wird.

MEDIENEREIGNISSE (I)

Ereignis	Wird ausgelöst...
onabort	wenn das Laden von Medieninhalten abgebrochen wird.
oncanplay	wenn eine Datei fertig zum Abspielen ist (d.h. der Buffer bis zu einer bestimmten Größe geladen wurde)
oncanplaythrough	wenn eine Datei komplett im Buffer geladen ist und ohne Unterbrechungen bis zum Ende abgespielt werden kann.
ondurationchange	wenn sich die Laufzeit verändert.
onemptied	wenn etwas unvorhergesehenes passiert ist und die Datei nicht abspielbereit ist (zum Beispiel Abbruch der Internetverbindung).
onended	wen der Abspielvorgang das Ende der Datei erreicht hat.
onerror	wenn eine Fehler beim Laden einer Mediendatei oder eines Medienstreams augetreten ist.
onloadeddata	wenn Mediendaten geladen wurden.
onloadedmetadata	wenn Metadaten (Abspieldauer, Format, etc.) geladen wurden.
onloadstart	in dem Moment wenn eine Datei beginnt zu laden.
onpause	wenn der Nutzer oder ein Programm die Abspielung anhält.
onplay	wenn der Medieninhalt bereit zum Abspielen ist.
onplaying	wenn mit dem Abspielen gerade begonnen wurde.
onprogress	wenn der Browser die Medieninhalte lädt.

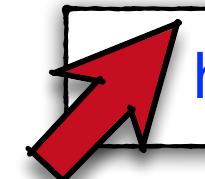


<https://developer.mozilla.org/en-US/docs/Web/API/Event>

5.2 EREIGNISSE EREIGNISTYPEN (III)

MEDIENEREIGNISSE (II)

Ereignis	Wird ausgelöst...
onratechange	jedesmal wenn die Abspielrate verändert wird.
onreadystatechange	wenn sich der "Ready State" einer Übertragung verändert.
onseeked	wenn das Seeking-Attribut auf false gesetzt wurde.
onseeking	wenn das Seeking-Attribut auf true gesetzt wurde.
onstalled	wenn der Browser den Inhalt nicht laden kann.
onsuspend	wenn das Laden von Inhalten vorzeitig abgebrochen wurde.
ontimeupdate	wenn sich die Abspielposition verändert hat.
onvolumechanged	wenn sich der Lautstärkepegel geändert hat.
onwaiting	wenn die Medienabspielung angehalten wurde, aber fortgesetzt werden soll (zum Beispiel bei Unterbrechung durch Leerlaufen des Buffer).



<https://developer.mozilla.org/en-US/docs/Web/API/Event>

5.2 EREIGNISSE

EREIGNISSE ZUM SELBERMACHEN (I)

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Hide And Seek</title>
5     <link href="style.css" rel="stylesheet" />
6     <meta charset="utf-8">
7   </head>
8   <body>
9     <h2>Erstes Geheimnis</h2>
10    <div>
11      <p>Das Schönste, was wir erleben können, ist das Geheimnisvolle. (Albert Einstein)</p>
12    </div>
13    <h2>Zweites Geheimnis</h2>
14    <div>
15      <p>Eine Frau ohne Geheimnisse ist wie eine Blume ohne Duft. (Maurice Chevalier)</p>
16    </div>
17    <h2>Drittes Geheimnis</h2>
18    <div>
19      <p>Das Geheimnis eines glücklichen Lebens liegt in der Entschuldigung. (Mahatma Gandhi)</p>
20    </div>
21    <script src="js/hideandseek.js"></script>
22  </body>
23 </html>
```

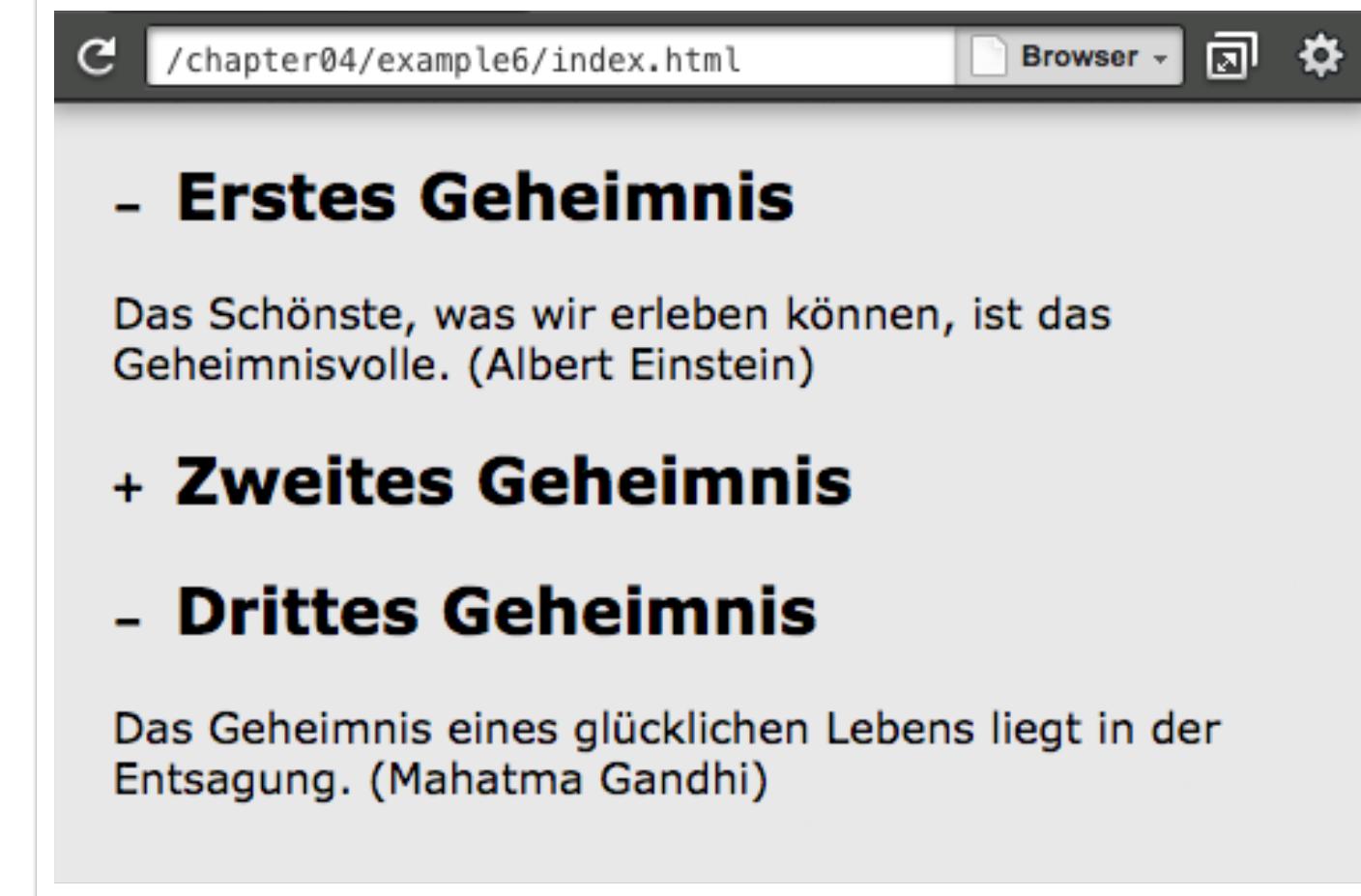
 /chapter05/example02/

5.2 EREIGNISSE

EREIGNISSE ZUM SELBERMACHEN (II)

```
js/hideandseek.js

1 var titles = document.getElementsByTagName('h2');
2
3 for (var i = 0; i < titles.length; i++) {
4     var title = titles[i];
5     title.innerHTML='<span class="mono">- </span>' + title.innerHTML;
6     title.addEventListener("click", beimKlicken);
7     var evt = new MouseEvent("click");
8     title.dispatchEvent(evt);
9 }
10
11 function beimKlicken(evt) {
12     var headerClicked=evt.currentTarget;
13     var relatedDiv=headerClicked.nextElementSibling;
14     var collapseMark=headerClicked.firstChild;
15     var isCollapsed=collapseMark.innerText[0]=="+";
16     collapseMark.innerText=isCollapsed ? "- " : "+ ";
17     relatedDiv.setAttribute("style", isCollapsed ? "" :
18     "display: none");
19 }
```



FORTSETZUNG FOLGT...



VORLESUNG

Prof. Dr. Axel Küpper

TU Berlin | T-Labs | Fachgebiet Service-centric Networking
Ernst-Reuter-Platz 7 | 10587 Berlin | Germany

 axel.kuepper@tu-berlin.de

 <https://twitter.com/kuepp>

 <https://www.linkedin.com/in/axelkuepper/>

 <http://www.snet.tu-berlin.de/kuepper>

ÜBUNGSLEITER

- Thomas Cory
- Sanjeet Raj Pandey
- Christian René Sechting

TUTOREN

- Nastassia Lukyanovich
- Maximilian Oliver Fisch
- Leonhardt Frederik Hollatz
- Adrian Siebing