Es war einmal ein kleiner unscheinbarer <video> -Tag…

HTML5 → Video

<video> as easy as <img>, so lautet die Botschaft.

- Manipulierbar durch JavaScript.
- Nutzbar für Spielereien wie rotierende oder zoomende Video-Fenster, aber auch **zur script-gesteuerten Kontrolle von Videos**.

Beispiel:

```
<video src="movie.mp4" controls="controls">
    your browser does not support the video tag
</video>
```

Tipp: Es ist möglich Text zwischen die Tags zu schreiben, der nur angezeigt wird, wenn der Browser HTML5 nicht darstellen kann.

HTML5 → Video

Die Attribute des Video-Tags

Attribute	Value	Beschreibung
autoplay	autoplay	Das Video wird sofort abgespielt.
controls	controls	Die Steuerelemente werden dargestellt.
height	*pixels*	Die Höhe des Videoplayers in der HTML-Seite.
width	*pixels*	Bestimmt die Breite des Videoplayers.
preload	preload	Das Video wird in die Seite geladen und ist bereit zum abspielen. Das Attribut wird ignoriert, wenn autoplay gesetzt ist.
src	*url*	Die URL des videos.
loop	loop	Das Video wird, sobald es am Ende ist, wieder gestartet.
muted	muted	Regelung der Lautstärke des Videos.
Poster	*URL*	Die URL eines Bildes, das während das Video geladen wird an dessen Stelle sichtbar ist.

Tutorials und Referenz zu HTML5 im Web: http://www.w3schools.com

HTML5 → Video

Das poster-Attribut

Falls ein Video nicht verfügbar sein sollte, kann als Ersatz ein statisches Bild angezeigt werden.

```
<video src="video.mp4" width="427" height="240" controls="controls"
    poster="platzhalter.jpg">
</video>
```

HTML5 → Video: DOM: Attribute

Playback Attribute:

- *currentTime*: Anzeige abgespielte Zeit
- *startTime*: Die Startzeit des Videos (Streams)
- *duration*: Laufzeit in sec
- *paused*: Wurde das Video angehalten?
- *ended*: Ist das Video zu Ende?
- *autoplay*: Ist autoplay vorhanden?

- *loop, autobuffer, seeking, defaultPlaybackRate, playbackRate, seekable, buffered, played, volume, muted, readyState, networkState, error*

```
function springeZu() {
   var video = document.getElementById('video');
        video.currentTime = 20;
    // Springe zur zwanzigsten Sekunde
}
```

Übung 2: HTML5 → Video → JavaScript-API

Filme einbinden und Schaltflächen anlegen

```
<body>
   <video src="BigBucktheora.mp4" id="video" height="400" width="600"></video>
<p>
   <button id="start" onClick="start()">Start/Pause</button>
   <button id="stumm" onClick="stumm()">Stummschalten</button>
   <button id="lauter" onClick="lauter()">Lauter</button>
   <button id="leiser" onClick="leiser()">Leiser</button>
</p>
</body>
```

Start/Pause | Stummschalten | Lauter | Leiser

Übung 2: HTML5 → Video → JavaScript-API

Javascript → Start/Pause

```
<script type="text/javascript">

var video = document.getElementById('video');

function start() {
     if (video.paused == true) {
           video.play();
     }
     else {
           video.pause();
     }
 }
```

Übung 2: HTML5 → Video → JavaScript-API

JavaScript → Stumm schalten

```
function stumm() {
     if(video.muted) {
          video.muted = false;
     }
     else {
      video.muted = true;
     }
 }
```

Übung 2: HTML5 → Video → JavaScript-API

Javascript → Lautstärkeregelung

```
 function lauter() {
     if(video.volume < 1) {
          video.volume = video.volume + 0.2;
     }
 }
 function leiser() {
     if(video.volume > 0) {
          video.volume = video.volume - 0.2;
     }
 }
</script>
```

Übung 2: HTML5 → Video → Navigation des Videos

Die Schaltfläche und das Textfeld für die Zeiteingabe:

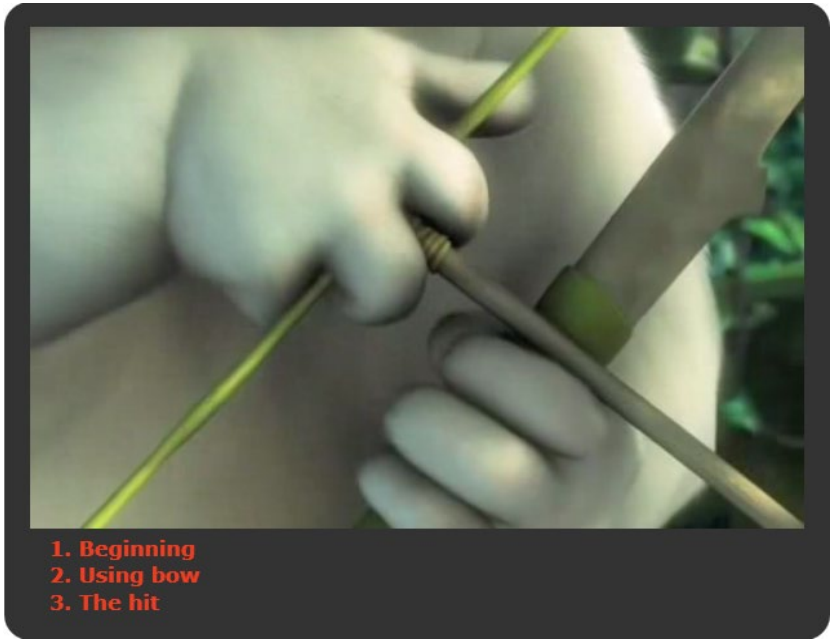
```
Zeit: <input name="" type="text" id="zeit"/>
      <button id="springe" onclick="springeZu()">Springe zu</button>
```

Die Funktion springeZu(), die zur gewünschten Sequenz führt:

```
function springeZu() {
 video.currentTime = document.getElementById('zeit').value;
}
```

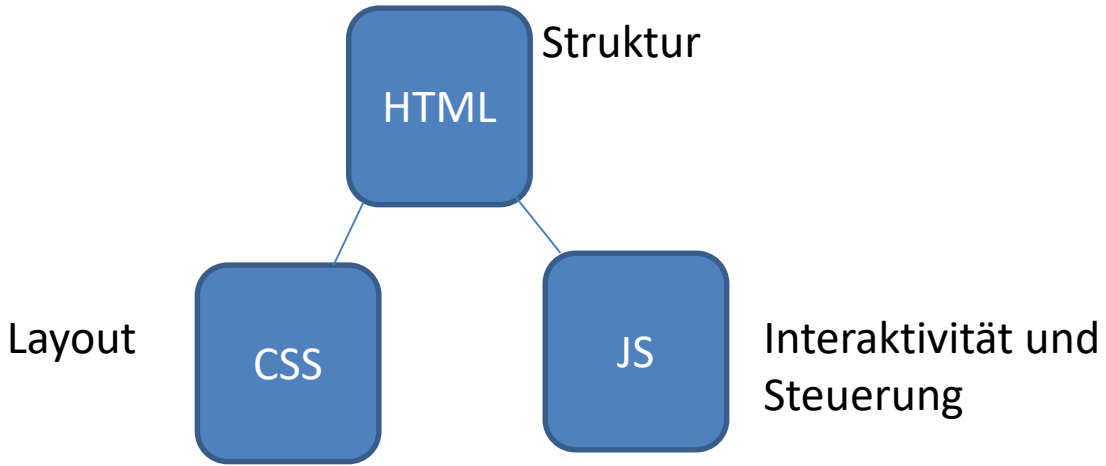
Übung 3: HTML5 → Video

Navigation des Videos über Text-Links



HTML5 = HTML + CSS + JavaScript

Wir bauen die kleine Anwendungen aus drei Dateien zusammen.
Die Dateien werden in der HTML-Datei miteinander verlinkt.



Übung 3: HTML5 → Video

Navigation des Videos über Text-Links

```
<div id="container">

<video src="BigBucktheora.mp4" id="film1" autoplay="autoplay" controls="controls" >
</video>

<a href="#" onClick="springeZu(0)">1. Beginning </a><br />
<a href="#" onClick="springeZu(4)">2. Using bow </a><br />
<a href="#" onClick="springeZu(24)">3. The hit </a><br />
</div>
```

Übung 3: HTML5 → Video

ACHTUNG!

```
<meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
```

Diese Zeile muss im “header” der HTML-Datei vorhanden sein, damit “Mediaqueries” funktionieren!

Der Viewport der mobilen Geräte ist meistens größer als der Screen. Hier zwingt das HTML Meta-Tag mit name="viewport" den Viewport auf Screengröße.
width=device-width heißt "Nicht zoomen!". Der Rest ist optional.
meta name="viewport" content="width=device-width" setzt die Breite des Screen bei mobilen Geräten auf die tatsächliche Auflösung in der Breite. Meta content="viewport" setzt die Breite auf die tatsächliche Breite. *initial-scale* verhindert das Zoomen beim Laden der Seite. Am Ende verhindert **user-scalable="no",** dass der Benutzer die Seite vergrößert oder verkleinert.

Übung 3: HTML5 → Video

Die CSS-Datei für das Layout:

```
@charset "utf-8";
/* CSS Document */

#container {
  margin: 0 auto;                  /* Container mittig setzen */
  background-color: #333333;       /* Hintergrundfarbe */
  padding: 20px;                   /* Abstand des Inhalts zum Rand */
  width: 480px;
  color: #EE3F20;
  border-radius: 20px;          /* Eckenradiusangabe für Webkit-Browser  */
}
```

Übung 3: HTML5 → Video

Navigation des Videos über Text-Links

Verlinkung der CSS-Datei in der HTML-Datei:

```
<head>
<title>Demo: HTML5 video controls with JavaScript</title>

<link href="example03.css" rel="stylesheet" type="text/css">

</head>
```

Übung 3: HTML5 → Video

Navigation des Videos über Text-Links

Externe JavaScript-Datei:

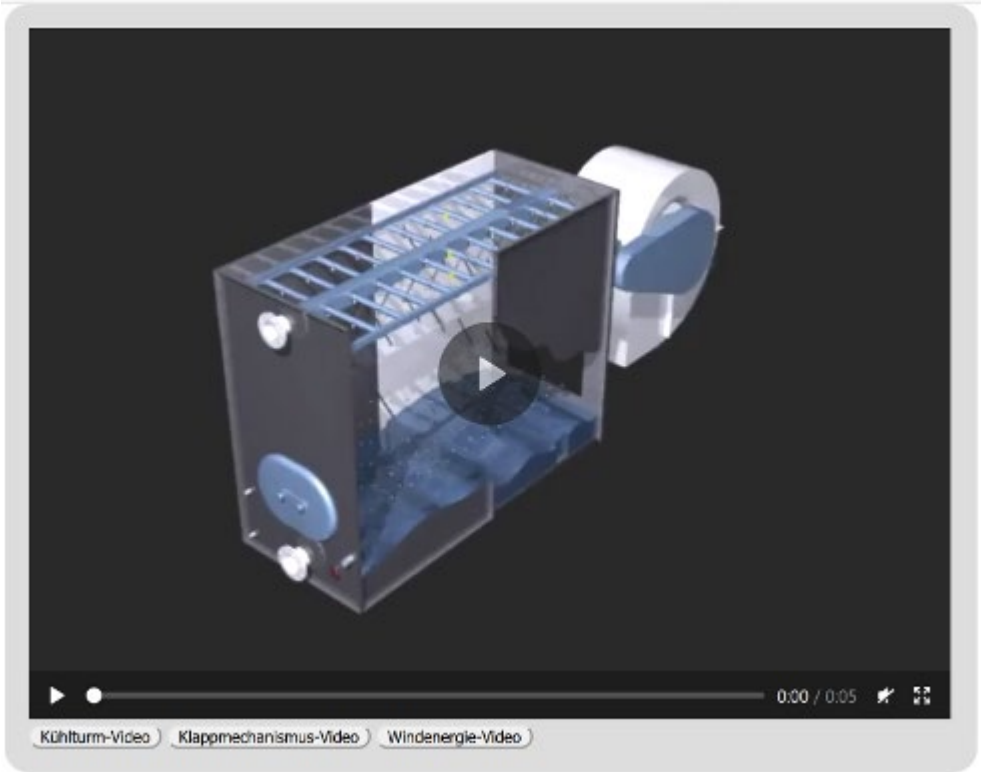
```
var video = document.getElementById('film1');

function springeZu(zeit) {
           video.currentTime = zeit;
}
```

Verlinkung der JS-Datei in der HTML-Datei:

```
<script src="example03.js" type="text/javascript"></script>
```

Übung 4: HTML5 → Video

Übung zur Auswahl von Videos



Übung 4: HTML5 → Video

Übung zur Auswahl von Videos

1. Das Video einbinden

```
<video id="film1" src="kuehlturm_720_576.mp4" controls height="480" width="640"></video>
```

ACHTUNG!

```
<meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no">
```

Diese Zeile muss im “header” der HTML-Datei vorhanden sein, damit “Mediaqueries” funktionieren!

Der Viewport der mobilen Geräte ist meistens größer als der Screen. Hier zwingt das HTML Meta-Tag mit name="viewport" den Viewport auf Screengröße.
width=device-width heißt "Nicht zoomen!". Der Rest ist optional.
meta name="viewport" content="width=device-width" setzt die Breite des Screen bei mobilen Geräten auf die tatsächliche Auflösung in der Breite. Meta content="viewport" setzt die Breite auf die tatsächliche Breite. *initial-scale* verhindert das Zoomen beim Laden der Seite. Am Ende verhindert **user-scalable="no",** dass der Benutzer die Seite vergrößert oder verkleinert.

Übung 4: HTML5 → Video:

Übung zur Auswahl von Videos

2. Auswahl der Filme und die Funktion des Ladens des ausgewählten Films.

Der Button-Tag mit den Auswahloptionen.

```
<button>Kühlturm-Video</button>
<button>Klappmechanismus-Video</button>
<button>Windenergie-Video</button>
```

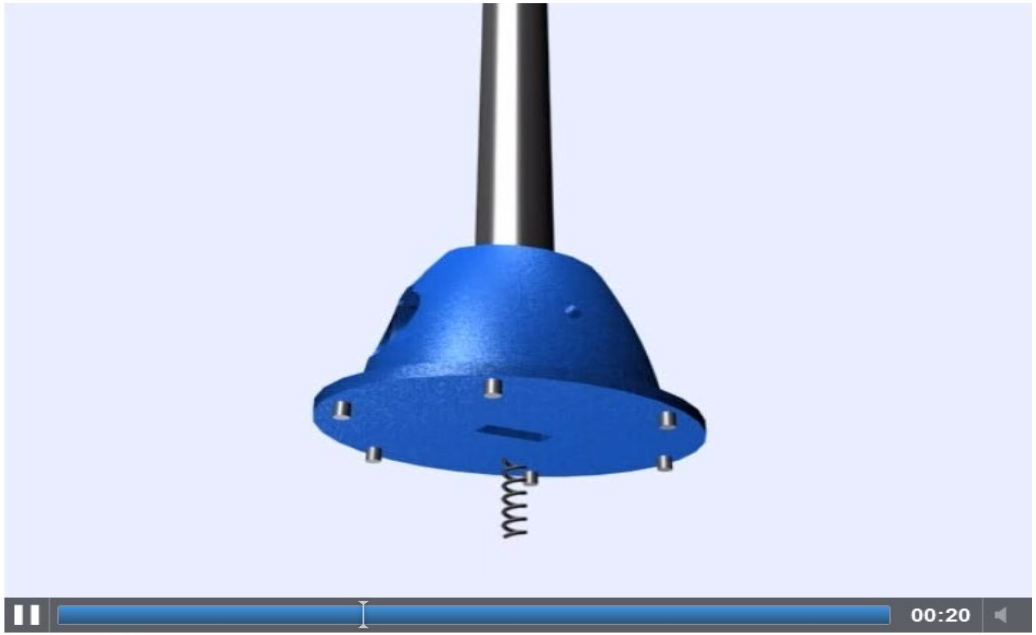
Übung 4: HTML5 → Video

Übung zur Auswahl von Videos

3. Funktion zum Wechseln des Videos.

```
document.getElementsByTagName("button")[1].onclick = function() {
                    film = "tisch.mp4";
                    video.src = film;
                    video.play();
          }
```

Übung 5: HTML5 → Video: Videofilme automatisch nachladen



Übung 5: HTML5 → Video: Videofilme automatisch nachladen

1. Ein Videofilm wird mit dem Video-Tag eingebunden.

2. Eine Funktion zum Erkennen des Filmendes mit der addEventListener()-Methode

3. Eine Funktion um den neuen Film zu laden

```
<body>

    <video controls src="tisch_720_576.mp4">
    </video>
    <br />
    <button onClick="myNewSrc()">nächstes Video</button>

</body>
```

Übung 5: HTML5 → Video: Videofilme automatisch nachladen

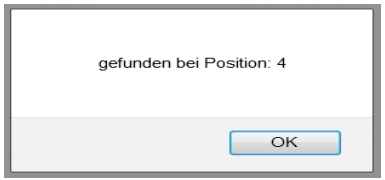
```
var myVideo = document.getElementsByTagName('video')[0];
myVideo.addEventListener('ended', myNewSrc, false);


function myNewSrc() {

        if (myVideo.src.indexOf("kuehlturm_720_576.mp4") != -1){
              myVideo.src="tisch_720_576.mp4";
        }
        else{
             myVideo.src="kuehlturm_720_576.mp4";
        }
        myVideo.load();
        myVideo.play();
    }
```

Übung 5: HTML5 → Video

Erklärung: indexOf()

Ermittelt das erste Vorkommen eines Zeichens oder einer Zeichenkette innerhalb einer Zeichenkette und gibt zurück, an wievielter Stelle das Zeichen in der Zeichenkette steht. Die Zählung beginnt bei 0. Wenn die Suche erfolglos ist, wird -1 zurückgegeben. Optional ist es möglich, die Funktion in einem zweiten Parameter anzuweisen, ab der wievielten Stelle in der Zeichenkette sie mit der Suche beginnen soll.

```
var Aussage = "Der Mensch ist dem Mensch sein Feind";
var Suche = Aussage.indexOf("Mensch");
alert("gefunden bei Position: " + Suche);
```



Übung 5: HTML5 → Video

Erklärung: indexOf()

Vorkommnisse prüfen

Beachten Sie dass '0' nicht als true gewertet wird und '-1' nicht als false gewertet wird. Deshalb wird wie folgt geprüft, ob eine bestimmte Zeichenfolge innerhalb einer anderen Zeichenfolge vorhanden ist:

```
'Blue Whale'.indexOf('Blue') != -1; // true
'Blue Whale'.indexOf('Bloe') != -1; // false
```

Übung 5: HTML5 → Video: addEventListener() und DOM-Events

Um einem Element die Events onmouseover, onmouseout oder onclick zuzuweisen, kann man auch die addEventlistener()-Methode nutzen.

addEventlistener ➔ Ereignisüberwacher

Bsp.:

```
var video = document.getElementsByTagName('video')[0];

video.addEventListener('mouseover', function() {
                    video.play();
               }, false);

video.addEventListener('mouseout', function() {
                    video.pause();
                    video.currentTime = 0;
              }, false);
```

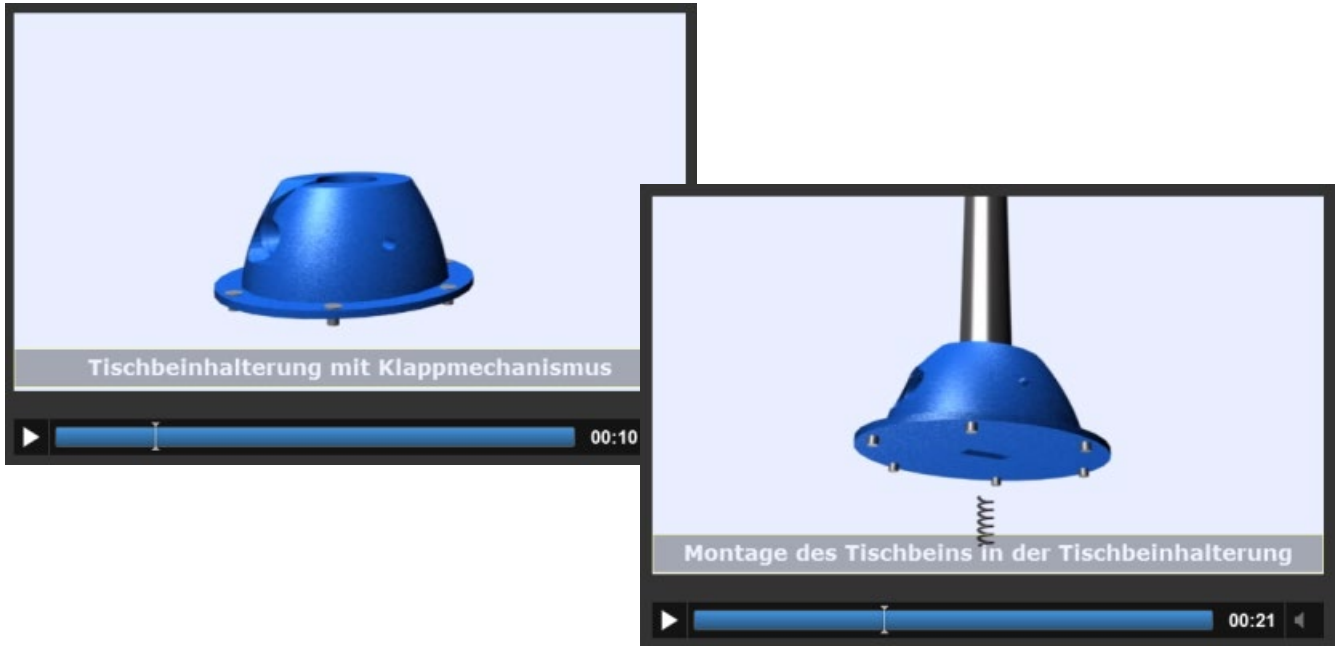
HTML5 → Video

Weitere DOM-Events des Video-Tags die mit der addEventListener-Methode kombiniert werden können.

ended	→	Das Ende der Datei ist erreicht.
abort	→	Der Browser hat den Download abgebrochen. Die Datei ist nicht vollständig heruntergeladen.
load	→	Die Datei wurde vollständig heruntergeladen.
loadeddata	→	Die ersten Daten der Datei können abgespielt werden.
progress	→	Die Datei wird heruntergeladen.
ratechange	→	Die Die Wiedergabegeschwindigkeit hat sich geändert.
timeupdate	→	Die aktuelle Position des Zeitstrahls hat sich geändert.

Übung 6: HTML5→Video

Zeitabhängig Texte einblenden



Übung 6: HTML5→Video - Zeitabhängig Texte einblenden

Der CSS-Abschnitt:

```
#Klappfuss {
    position:relative;
    padding:4px;
    width:590px;
    height:25px;
    z-index:2;
    left: 0px;
    top: -100px;
    font-family: Verdana, Geneva, sans-serif;
    font-size: 14pt;
    color: #FFF;
    visibility: hidden;
    font-weight: bold;
    background-color: #000000;
    text-align:center;
    border: thin solid #FF0;
    opacity: 0.3;
}
```

Übung 6: HTML5→Video - Zeitabhängig Texte einblenden

Der HTML-Teil:

```
<body>
<div id="container">
 <video id="video" src="tisch_720_576.mp4" width="600px" height="450px" controls autoplay>
</video>
  <div id="Klappfuss">Tischbeinhalterung mit Klappmechanismus</div>
  <div id="Montage">Montage des Tischbeins in der Tischbeinhalterung</div>
</div>
</body>
```

Übung 6: HTML5→Video - Zeitabhängig Texte einblenden

Der JavaScript-Teil: 1 von 2

```
<script>

var myVideo = document.getElementById('video');

function myNewSrc() {
   if ((myVideo.currentTime > 5) && (myVideo.currentTime < 15))
   { document.getElementById("Klappfuss").style.visibility="visible";
   }
   else { document.getElementById("Klappfuss").style.visibility="hidden";
   }
   if ((myVideo.currentTime > 18) && (myVideo.currentTime < 35)) {
      document.getElementById("Montage").style.visibility = "visible";
   }
   else {document.getElementById("Montage").style.visibility = "hidden" ;
   }
}
```

Übung 6: HTML5→Video - Zeitabhängig Texte einblenden

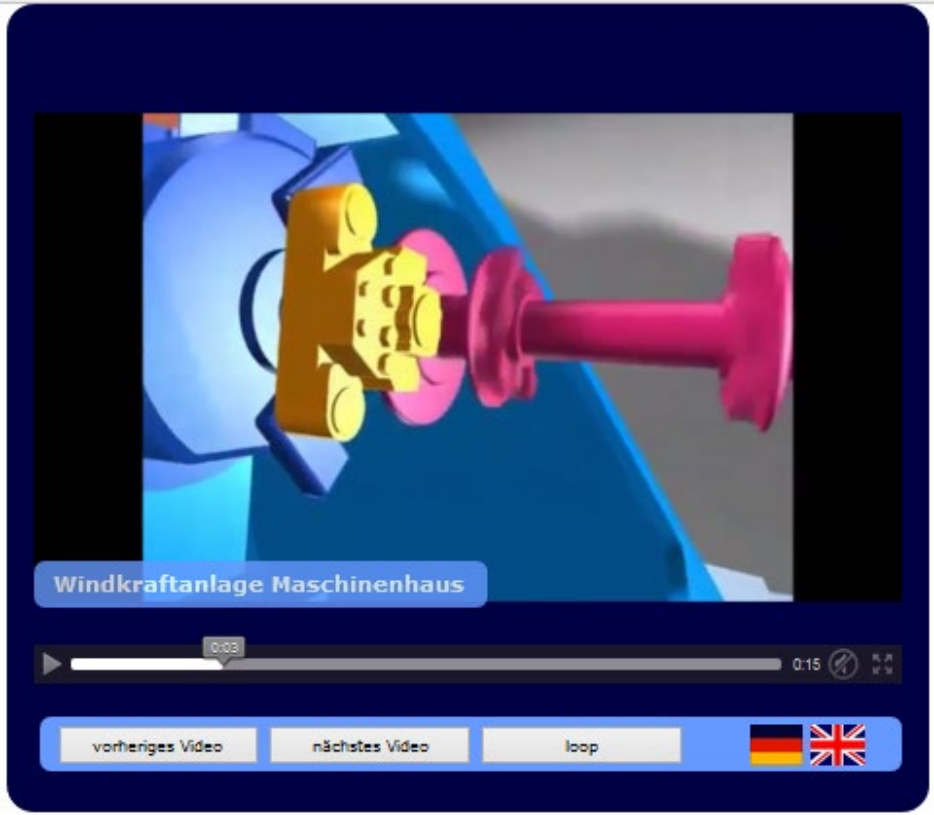
Der JavaScript-Teil: 2 von 2

```
myVideo.addEventListener('timeupdate',myNewSrc, false);

</script>
```

timeupdate → Die aktuelle Position des Zeitstrahls hat sich geändert.

Übung 8: HTML5→Video

Interaktive,
mehrsprachige
Videoanleitung



Übung 8: XML-Datei in Deutsch

```
<?xml version='1.0' encoding='utf-8'?>
<films>
 <filme>3</filme>
 <filmtyp>.ogv</filmtyp>
 <filmtyp>.mp4</filmtyp>
 <filmname>Windenergie</filmname>
 <filmtexte>Windenergieanlage</filmtexte>
 <filmtexte>Windkraftanlage Maschinenhaus</filmtexte>
 <filmtexte>Motor und Generator</filmtexte>
 <button>vorheriges Video</button>
 <button>nächstes Video</button>
 <startheader>Windenergieanlage:</startheader>
 <starttext>In den drei Filmsequenzen wird eine Windenergieanlage
    vorgestellt.Sie können mit den Schaltflächen die Filmsequenzen
    vorwärts und rückwärts durchgehen. Mit einem Klick auf die
    jeweilige Flagge können Sie die Sprache umstellen.</starttext>
</films>
```

Übung 8: XML-Datei in Englisch

```
<?xml version='1.0' encoding='utf-8'?>
<films>
 <filme>3</filme>
 <filmtyp>.ogv</filmtyp>
 <filmtyp>.mp4</filmtyp>
 <filmname>Windenergie</filmname>
 <filmtexte>Wind energy plant</filmtexte>
 <filmtexte>Machinery housing</filmtexte>
 <filmtexte>Gearbox and Generator</filmtexte>
 <button>previous video</button>
 <button>next video</button>
 <startheader>Wind Power Plant:</startheader>
 <starttext>In the three movies a wind turbine is presented. You can play
    all three sequences by clicking the forward or backward buttons. By
    clicking on the flag you can change the language.</starttext>
</films>
```

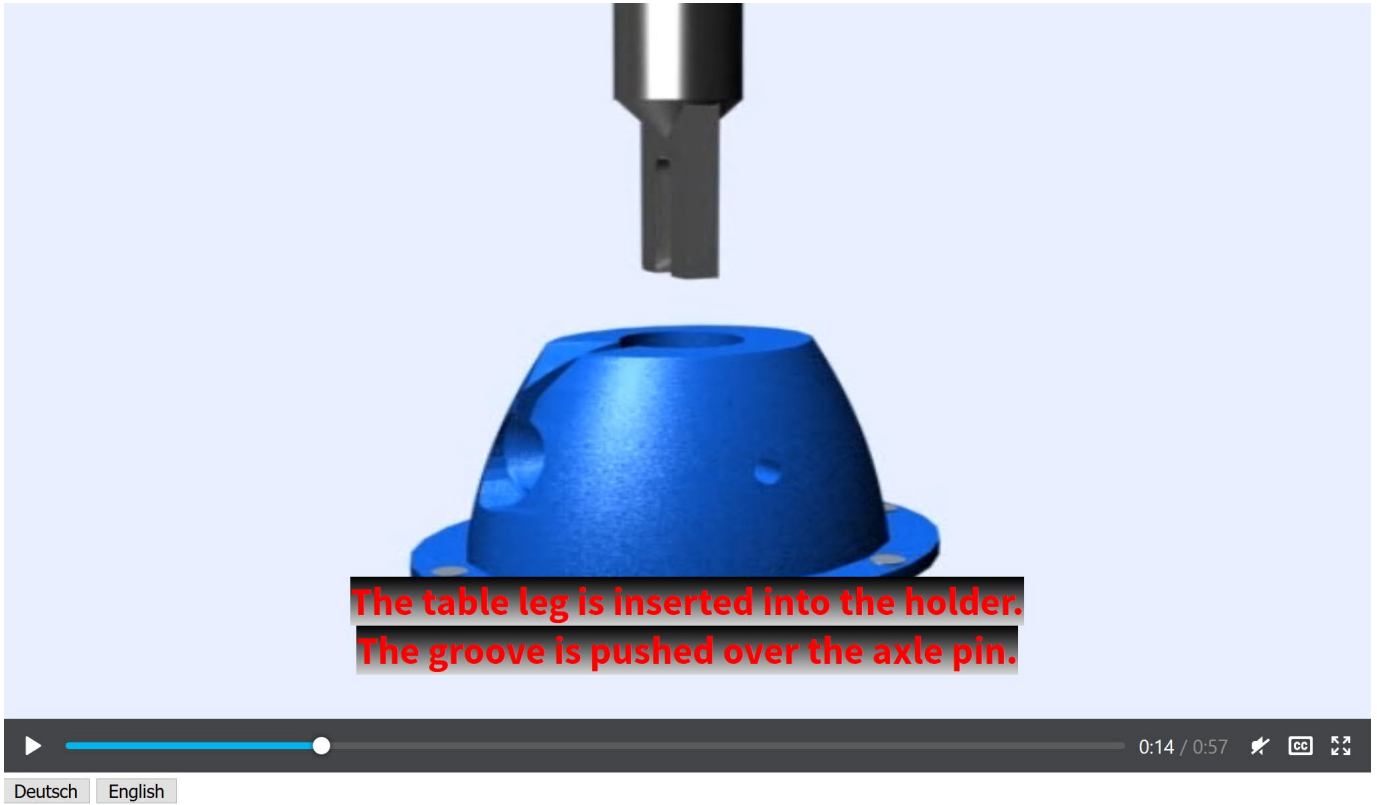
Übung 8: AJAX-Technologie zum Auslesen von XML-Daten

```
sprachauswahl("example06_deu.xml");

function sprachauswahl(sprach_datei) {
  xmlhttp.open("GET", sprach_datei, true);
…
var xmldoc = xmlhttp.responseXML;
       Filmanzahl = xmldoc.getElementsByTagName('filme').item(0).firstChild.data;

       Filmtyp1   = xmldoc.getElementsByTagName('filmtyp').item(0).firstChild.data;
       Filmtyp2   = xmldoc.getElementsByTagName('filmtyp').item(1).firstChild.data;
       Filmname   = xmldoc.getElementsByTagName('filmname').item(0).firstChild.data;
       texte = new Array();
          for (k=0;k<Filmanzahl;k++) {
            texte[k] = xmldoc.getElementsByTagName('filmtexte').item(k).firstChild.data;
          }
…
```

Übung 9: HTML5 → Audio

```
<!DOCTYPE HTML>
<html>
<body>

<button type="button" id="playbutton">Click Me!</button>

</body>

<script type="text/javascript">
   document.getElementById('playbutton').onclick = function() {
            new Audio('allorge.ogg').play();
   }
</script>
</html>
```

Übung 15: TextToSpeech mit ResponsiveVoice.js und webVTT



Übung 15: HTML: Der Track-Tag

```
<video id="film" src="tisch_720_576.mp4" controls>
  <track default mode="2" kind="subtitles" label="English
   subtitles" src="eng.vtt" srclang="en" ></track>
</video>

<br>
<button id="d">Deutsch</button>
<button id="e">English</button>
```

textTracks[0]

Übung 15: Die webVTT-Dateien (Deutsch)

startTime

endTime

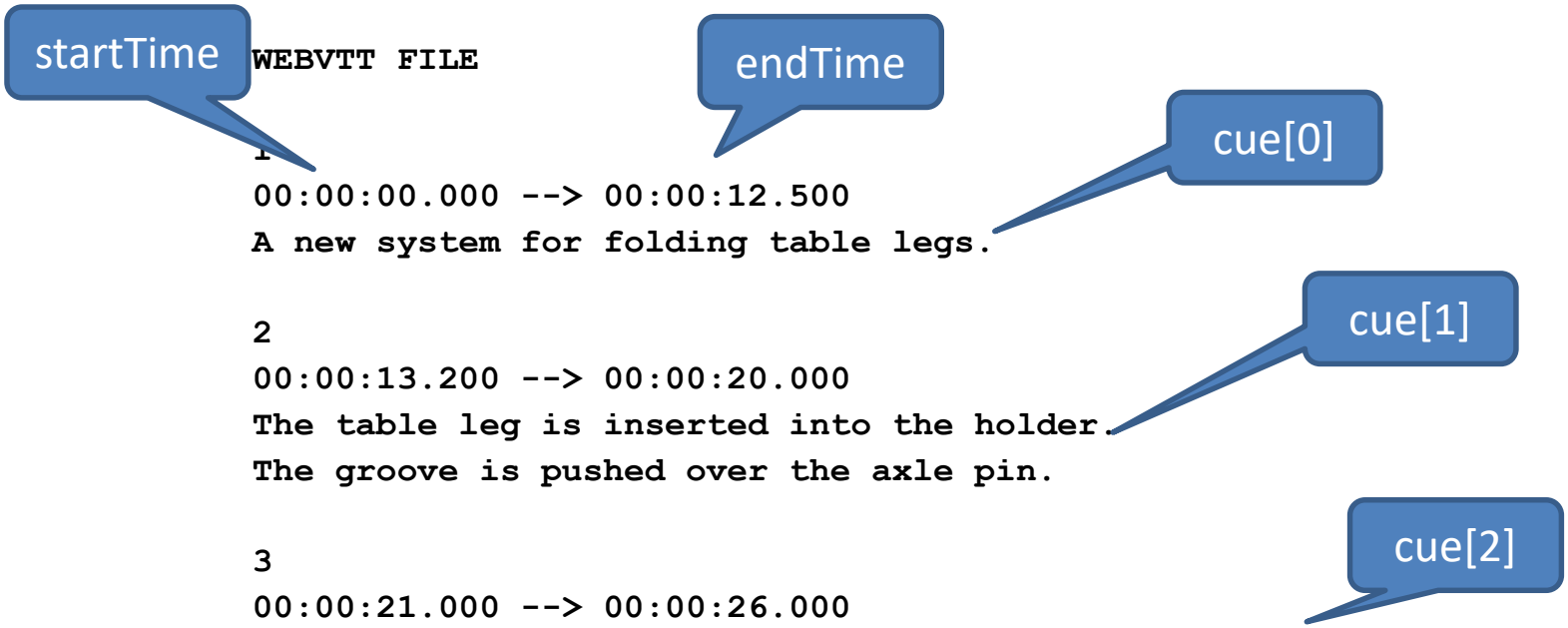
cue[0]

cue[1]

cue[2]

```
WEBVTT FILE

1
00:00:00.000 --> 00:00:12.500
Ein neues System für knickbare Tischbeine.

2
00:00:13.000 --> 00:00:20.000
Das Tischbein wird in die Halterung eingeführt.
Der Spalt am Ende des Tischbeins wird über den Achsstift
geschoben.

3
00:00:20.000 --> 00:00:26.000
Von unten wird eine Feder eingeführt und mit einem
Abschlussstück fixiert.
```

Übung 15: Die webVTT-Dateien (Englisch)



Übung 15: JavaScript: Variablen und Ob

```
var e = 0; // Zählvariable für Sprachzeilen (cues) zu zählen
var myVideo = document.getElementById('film');
var myTrack = document.getElementsByTagName("track")[0];
```

Methode um den Tag „video“ in der Webseite anhand der id „film“ zuzuweisen

Methode um den Tag „track“ anhand der Position des tracks in der Webseite zuzuweisen

Übung 15: JavaScript: Sprachumschaltung Deutsch über Button-Klick

```
document.getElementById("d").onclick = function() {

  myTrack.label = "Deutsche Untertitel";
  myTrack.src = "deu.vtt";
  myTrack.srclang = "de";

  e=0;
  myVideo.currentTime = 0;
```

Methode um den Tag „button“ in der Webseite anhand der id „d“ auf einen Klick zu überprüfen.

Zuweisung der deutschen vtt-Datei zum track und damit die Initialisierung der Sprachumschaltung.

Die Zählvariable e wird auf 0 gesetzt. Damit beginnt die Funktion wieder mit der ersten Zeile (cue) der Textdatei.

Wenn die neue Sprache zugeschaltet wird, wird das Video auf den Anfang (0s) gesetzt.

Übung 15: JavaScript: Sprachumschaltung Englisch über Button-Klick

```
document.getElementById("e").onclick = function() {

  myTrack.label = "English subtitles";
  myTrack.src = "eng.vtt";
  myTrack.srclang = "en";

  e=0;
  myVideo.currentTime = 0;
```

Methode um den Tag „button“ in der Webseite anhand der id „e“ auf einen Klick zu überprüfen.

Zuweisung der englischen vtt-Datei zum track und damit die Initialisierung der Sprachumschaltung.

Wenn die neue Sprache zugeschaltet wird, wird das Video auf den Anfang (0s) gesetzt.

Die Zählvariable e wird auf 0 gesetzt. Damit beginnt die Funktion wieder mit der ersten Zeile (cue) der Textdatei.

Übung 15: JS: Kontrolle der Filmzeit, um Texte zur korrekten Zeit vorzulesen

```
function textChange() {

    if ((e < myVideo.textTracks[0].cues.length) &&
       (myVideo.currentTime > myVideo.textTracks[0].cues[e].startTime)){
        var talk = myVideo.textTracks[0].cues[e].text;

        if (myTrack.srclang === "en")
            responsiveVoice.speak(talk, "UK English Male");
        else responsiveVoice.speak(talk, "Deutsch Male");

        e++;
    }
}

myVideo.addEventListener('timeupdate',textChange, false);
```

Anzahl der Textzeilen in der vtt-Datei

Text der aktuellen cue

Startzeit der aktuellen cue

(Überwachungs-)Methode, um nach jeder Zeitänderung des Videos die Funktion **textchange** aufzurufen.

Prof. Dipl.-Ing. Martin Schober | Informations- und Medientechnik

Studiendekan des Master-Studiengangs KMM
Projektpartner Shells: Shared Excellence – Laboratory Learning Spaces 4.0
Projektpartner EVEIL3D - Lernen in virtuellen Welten
Tagungsbeirat tekom e.V.
Multimedia-AG

Hochschule Karlsruhe - Technik und Wirtschaft
Fakultät für Informationsmanagement und Medien

Postanschrift: Postfach 24 40, 76012 Karlsruhe
Besucheranschrift: Amalienstr. 81-87 | 76133 Karlsruhe | Raum AM 113
fon +49 (0)721 925 - 2990 | fax +49 (0)721 925 - 1125
mobil +49 (0)173 945 82 18
martin.schober@h-ka.de

technischeredaktion.com/multimediaprojekte