HTML5, JavaScript, CSS3

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.ogv" 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/

HTML5 → Video *für alte Browser*

Der Source-Tag erlaubt mehrere Medienquellen für ein Element

Der Source-Tag definiert Medienresourcen wie <video> und <audio>.

```
<video controls="controls" autoplay="autoplay">
    <source src="RollerFilm1.mp4"
      type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"' />
   // Safari, IE9, Chrome
    <source src="RollerFilm1.ogv"
      type='video/ogg; codecs="theora, vorbis"' />
    // Mozilla, Firefox, Opera, Chrome
</video>
```

Attribute	Wert	Beschreibung
media	*media_query*	Definiert den Typ der Medienquelle , um zu Entscheiden ob das Medium herunter geladen wird oder nicht.
src	*url*	Die URL der Medien.
type	*MIME-Type*	Definiert den MIME-Type des Mediums.

HTML5 → Video *für neue Browser*

Der Source-Tag erlaubt mehrere Medienquellen für ein Element

Der Source-Tag definiert Medienresourcen wie <video> und <audio>.

```
<video src="RollerFilm1.mp4" controls="controls" autoplay="autoplay">
<!-- Safari, IE9, Chrome Mozilla, Firefox, Opera, Chrome -->
</video>
```

Attribute	Wert	Beschreibung
media	*media_query*	Definiert den Typ der Medienquelle , um zu Entscheiden ob das Medium herunter geladen wird oder nicht.
src	*url*	Die URL der Medien.
type	*MIME-Type*	Definiert den MIME-Type des Mediums.

HTML5 → Video

Dateiauswahl nach Media-Typ.

Ähnlich wie in CSS können Media_queries ausgewählt werden. Zum Beispiel einen bestimmten Film für eine Smartphone-Auflösung.

```
<video controls="controls" autoplay="autoplay">
    <source src="video_klein.ogg" media="handheld">
    <source src="video_normal.ogg" media="all">
</video>
```

Source wird der Reihe nach verarbeitet. Findet ein Gerät das für sich Passende, z.B. „handheld“, dann wird nicht weiter gesucht.

Media-Typen des W3C

Das W3C hat die folgenden Typen für media vorgesehen:

all	Für alle Geräte
braille	Braillezeile
embossed	Braille-Drucker
handheld	Geräte mit kleinen Monitoren
print	Drucker
projection	Präsenationen
screen	Desktop-Computer
speech	Sprachausgabe
tty	Geräte mit einer festen Platz pro Zeichen
tv	TV, keine oder nur begrenzte Möglichkeit zu Scrollen

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: Document Object Model: Attribute und Events

Video Attribute:

- *src*: stellt die URL der Videodatei zur Verfügung
- *poster (URL)*: ein Bild, das vom Browser angezeigt wird, solange das Video lädt.
- *controls*: Der Browser stellt die integrierten controls für Playback und Lautstärke zur Verfügung.
- *width*:
- *height:*

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 1: 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 1: 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 1: HTML5 → Video → JavaScript-API

JavaScript → Stumm schalten

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

Übung 1: 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 1: HTML5 → Video → Navigation des Videos

Die Schaltfläche und das Textfeld für die Zeiteingabe:

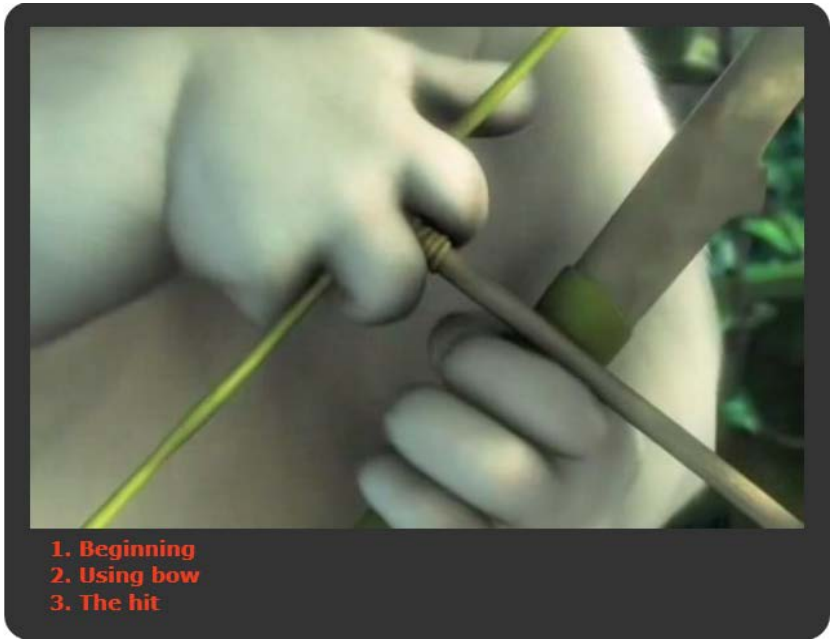
```
Zeit: <input name="" type="text" id="zeit"/>
      <button id="springe" onclick="springeZu()">Springe zu</button>
```

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

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

Übung 2: HTML5 → Video

Navigation des Videos über Text-Links



Übung 2: HTML5 → Video

Navigation des Videos über Text-Links

```
<div id="container">

<video src="BigBucktheora.mp4" id="video" 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 2: HTML5 → Video

Navigation des Videos über Text-Links

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: 600px;
  color: #EE3F20;
  -moz-border-radius: 20px; /* Eckenradiusangabe für Mozilla-Browser */
  border-radius: 20px;      /* Eckenradiusangabe für Webkit-Browser  */
}
```

Übung 2: 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 2: HTML5 → Video

Navigation des Videos über Text-Links

Externe JavaScript-Datei:

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

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

Verlinkung der JS-Datei in der HTML-Datei:

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

Übung 2: HTML5 → Neues Video einsetzen

1. Ersetzen Sie den Videofilm durch den Videofilm über die Brennstoffzelle.

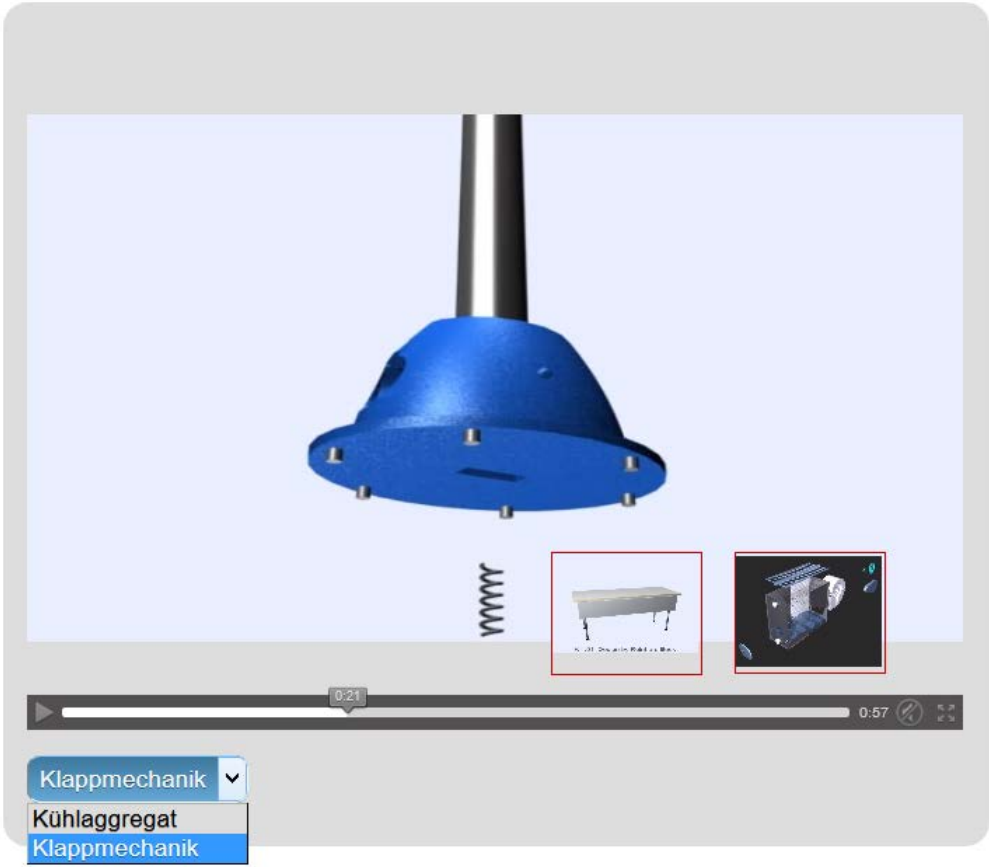
2. Setzen Sie neue Sprungmarken zu den Themen
    1. Start
    2. Grundlagen
    3. Prinzip Brennstoffzelle
    4. Wasserstoffgewinnung
    5. Elektrolyse

    Finden Sie heraus zu welcher Zeit das entsprechende Thema beginnt und geben Sie die entsprechenden Sekundenzahlen im Programm ein.

3. Sorgen Sie dafür, dass der Film immer gestartet wird, wenn ein Thema angeklickt wird. Der Befehl dafür lautet: **`video.play();`**

Übung 3: HTML5 → Video

Übung zur Auswahl von Videos



Übung 3: HTML5 → Video

Übung zur Auswahl von Videos

1. Das Video einbinden

```
<video id="video" src="kuehlturm_720_576.mp4" controls height="576" width="768"></video>
```

Übung 3: HTML5 → Video:

Übung zur Auswahl von Videos

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

Der Select-Tag mit den Auswahloptionen.


```
<select id="auswahl">
      <option value="kuehlturm_720_576.mp4">Kühlaggregat</option>
      <option value="tisch_720_576.mp4">Klappmechanik</option>
</select>
```

Übung 3: HTML5 → Video

Übung zur Auswahl von Videos

3. Funktion zum Wechseln des Videos.

```
document.getElementsByTagName("option")[0].onclick = function(){
    video.src = this.value;
    video.currentTime = 0;
    video.load();
   }
```

Übung 4: HTML5 → Video: Videofilme automatisch nachladen

Übung 4: 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 onLoad="myAddListener()">

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

</body>
```

Übung 4: HTML5 → Video: Videofilme automatisch nachladen

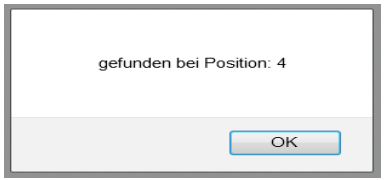
```
function myAddListener(){
        var myVideo = document.getElementsByTagName('video')[0];
        myVideo.addEventListener('ended', myNewSrc, false);
        }

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 4: 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 4: 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 4: 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);
```

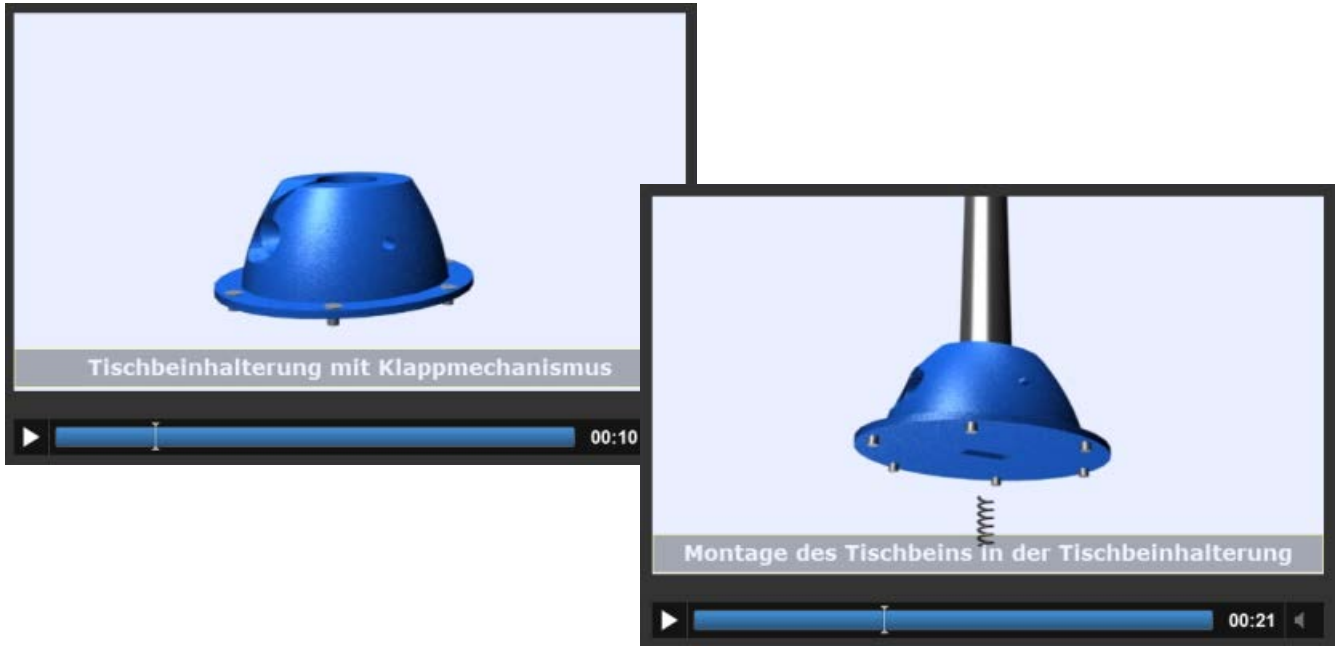
HTML5 → Video

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 5: HTML5→Video

Zeitabhängig Texte einblenden



Übung 5: 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 5: HTML5→Video - Zeitabhängig Texte einblenden

Der HTML-Teil:

```
<body onLoad="myAddListener()">
<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 5: HTML5→Video - Zeitabhängig Texte einblenden

Der JavaScript-Teil:

```
<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 5: HTML5→Video - Zeitabhängig Texte einblenden

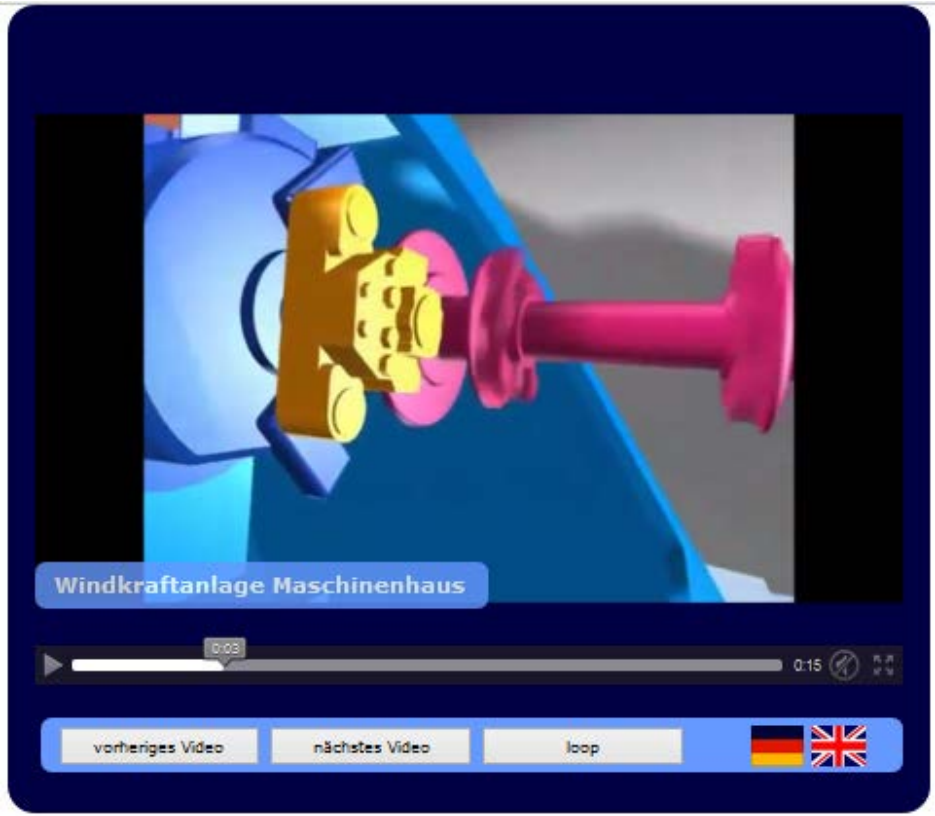
Der JavaScript-Teil:

```
function myAddListener(){
        myVideo.addEventListener('timeupdate',myNewSrc, false);
}

</script>
```

timeupdate → Die aktuelle Position des Zeitstrahls hat sich geändert.

Übung 6: HTML5→Video

Interaktive,
mehrsprachige
Videoanleitung



Übung 6: 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 6: 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>
```

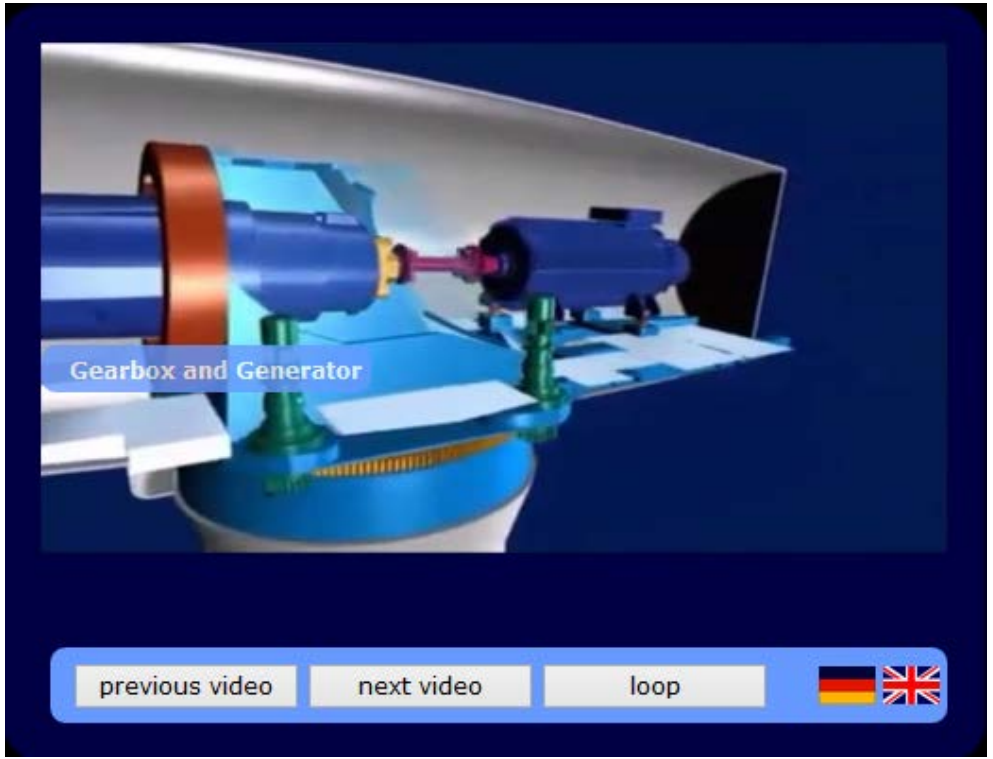
Übung 6: AJAX-Technologie zum Auslesen von XML-Daten

```
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 7: Interaktive und mehrsprachige Videoanleitungen

Der Video-Tag lässt Erweiterungen bezüglich Interaktivität und Steuerung von Videos mittels JavaScript-Programmierung zu.

1. Die Anwendung besteht aus vier Videosequenzen.
2. Alle Sprachdaten werden aus XML-Dateien geladen.
3. Interaktivität durch Aufruf der Sequenzen mit Klicks.
4. Die Sequenzen werden als Gesamtfilm abgespielt mit "Loop".
5. Sequenzenbezogene Texte werden eingeblendet.
6. Ein Link in Sequenz 2 führt zu einem Film über ein Detail (Generator).
7. Ein Link in Sequenz 3 führt zu einer URL über Windkraftanlagen.



Neuerungen in JavaScript die in Übung 6 berücksichtigt sind z. B.: “use strict“

Mit *use strict* aus dem ECMAScript 5-Standard zieht Disziplin ins Programmierstübchen. Für die neue Generation von Javascript-Anwendungen ist der liebevolle Umgang mit den kleinen Fehlern (fehlende Semikolons, undeklarierte Variablen, …) ein Handicap.

ECMAScript 5 im strict mode bringt eine leicht abweichende Semantik und fordert eine saubere Programmierung.

use strict meldet sich bei fehlenden Semikolons, verbietet *with*, unterwandert *eval* und reserviert neue Namen für spätere Versionen (package, private, protected, pbulic, static, yield).

Quelle: https://www.mediaevent.de/javascript/strict-mode.html

Neuerungen in JavaScript die in Übung 6 berücksichtigt sind z. B.: Vegleichsoperatoren

Operator	Beschreibung	Beispiel	Resultat	JavaScript
===	Prüft, ob die Aussagen gleich und vom gleichen Type sind	1 === 1 a === 1	true false	1.3
!==	Prüft, ob die Aussagen ungleich und nicht vom gleichen Type sind	1 !== 1 a !== 1	false true	1.3

Quelle: http://www.a-coding-project.de/ratgeber/javascript/operatoren

Übung 7: Links in Videosequenzen zeitabhängig einblenden

```
myVideo.addEventListener('timeupdate', function() {

    if (myVideo.src.indexOf("Windenergie2.mp4") != -1) {

         if ((myVideo.currentTime > 5) && (myVideo.currentTime < 15))
         { document.getElementById("filmlink").style.visibility = "visible";}
         else {
              document.getElementById("filmlink").style.visibility = "hidden";
         }
    }

     if (myVideo.src.indexOf("Windenergie3.mp4") != -1) {

         if ((myVideo.currentTime > 1) && (myVideo.currentTime < 10))
         { document.getElementById("urllink").style.visibility = "visible";}
         else {
              document.getElementById("urllink").style.visibility = "hidden";
         }
    }
```

Übung 7: Link ausblenden und neuen Film laden und abspielen

```
document.getElementById("filmlink").onclick = function() {
   document.getElementById("filmlink").style.visibility = "hidden";
    myVideo.src = "planeten_dem_Seq1.mp4";
    myVideo.load();
    myVideo.play();
    };
```

Übung 8: 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>
```