HTML5 APIS

JAVASCRIPT STORAGES, MEDIA, GEOLOCATION

LOCAL STORAGE API, SESSION STORAGE API, INDEXED DB API

WEB STORAGES - LOKALES SPEICHERN VON DATEN

- localStorage - Datensicherung ohne Zeitlimit
- sessionStorage - Datensicherung für eine Session
- In Folgenden werden die Methoden anhand des localStorage erläutert, der sessionStorage funktioniert analog.

- Bisher wurde das mit Cookies umgesetzt. Cookies eigenen sich aber nicht für das Handling von größeren Datenmengen, da sie bei JEDEM Request an den Server abgerufen werden. Das ist langsam und wenig effizient.

- HTML5 ruft die Daten nicht bei jedem Aufruf ab, sondern nur bei einer Abfrage. So stellt auch das Handling größerer Datenmengen keine Beeinträchtigung der Performance dar.

- Daten werden in einem eigenen, isolierten Bereich pro Domain gespeichert und sind diesem zugeordnet.
- Zum Speichern und Abrufen der gespeicherten Daten wird Javascript verwendet.

EINEN LOCALSTORAGE ERZEUGEN UND ABRUFEN:

```
<script type="text/javascript">
   localStorage.lastname="Smith";
   document.write(localStorage.lastname);
</script>
```

LOCALSTORAGE - FUNKTIONEN

```
// Speichern, lesen und löschen erfolgt über Javascript
// Funktionen und dem localStorage Objekt.

localStorage.setItem (key, value);
localStorage.getItem (key);
localStorage.removeItem (key);
```

EIN LOKALER SEITENZÄHLER

```
if (localStorage.pagecount)  {
   localStorage.pagecount
      =Number(localStorage.pagecount) +1;
} else {
   localStorage.pagecount=1;
}

document.write("Visits "
   + localStorage.pagecount
   + " time(s)."
);
```

FILE/BLOB

FILELIST

FILEREADER

BLOBBUILDER

FILEWRITER

DIRECTORYREADER

LOCALFILESYSTEM

EIN DATEISYSTEM ANFRAGEN

- window.requestFileSystem() fordert Zugriff auf ein durch die Sandboxing-Technologie geschütztes Dateisystem an:

```
window.requestFileSystem  = window.requestFileSystem ||
window.webkitRequestFileSystem;
```

- ```
  window.requestFileSystem(
   type,
   size,
   successCallback,
   opt_errorCallback
  );
  ```
  ```

TYPE: PERSISTENT ODER TEMPORARY

- window.TEMPORARY oder window.PERSISTENT.
- Daten unter TEMPORARY können vom Browser bei Bedarf entfernt werden, beispielsweise wenn mehr Speicherplatz benötigt wird.
- PERSISTENT Speicher kann nicht vom Browser gelöscht werden, nur vom Nutzer oder von der App.
- Für PERSISTENT muss vom Nutzer ein Kontingent eingeräumt werden.

SIZE

- Die von der Anwendung benötigte Größe (in Byte) für den Speicher.
- 5 * 1024 * 1024 => 5Mb

SUCCESSCALLBACK NACH DEN ANLEGEN DES SPEICHERS

- successCallback
  Callback, das bei einer erfolgreichen Anforderung eines Dateisystems aufgerufen wird. Das Argument ist ein FileSystem-Objekt.

ERRORCALLBACK IM FEHLERFALL

- Optionales Callback für die Behandlung von Fehlern oder bei der Ablehnung einer Anforderung zum Dateisystemabruf.
- Das Argument ist einFileError-Objekt.

BEISPIEL

- function onInitFs(fs) {
  console.log('Opened file system: ' + fs.name);
  }
  window.requestFileSystem(
  window.TEMPORARY, 5*1024*1024, onInitFs,
  errorHandler
  );

- function errorHandler(e) {
    var msg = '';
    switch (e.code) {
        case FileError.QUOTA_EXCEEDED_ERR:
            msg = 'QUOTA_EXCEEDED_ERR';
            break;
        case FileError.NOT_FOUND_ERR:
            msg = 'NOT_FOUND_ERR';
            break;
        case FileError.SECURITY_ERR;
            msg = 'SECURITY_ERR';
            break;
        case FileError.INVALID_MODIFICATION_ERR:
            msg = 'INVALID_MODIFICATION_ERR';
            break;
        case FileError.INVALID_STATE_ERR:
            msg = 'INVALID_STATE_ERR';
            break;
        default:
            msg = 'Unknown Error';
            break;
    };
    console.log('Error: ' + msg);
}

SPEICHER-KONTINGENT ANFORDERN

- PERSISTENTer Speicher erfordert eine Nutzerberechtigung.
- TEMPORARY Speicher braucht dies nicht, da der Browser die Daten bei Bedarf entfernen kann.
- Unter Chrome gibt es für PERSISTENTen Speicher unter webkitStorageInfo über ein neues API zur Anforderung von Speicher:

```
window.webkitStorageInfo.requestQuota(
  PERSISTENT, 1024*1024,
  function(grantedBytes) {
        window.requestFileSystem(
              PERSISTENT, grantedBytes, onInitFs, errorHandler
        );
  }, function(e) {
  console.log('Error', e);
});
```

- Nach einmaliger Berechtigung muss requestQuota() nicht mehr aufgerufen werden.
- Nachfolgende Aufrufe unterliegen einer NOOP-Anweisung (No Operation Performed), sodass kein Vorgang ausgeführt wird.
- Darüber hinaus gibt es ein API zur Abfrage der aktuellen Kontingentnutzung und -zuweisung des Ursprungs:

```
window
 .webkitStorageInfo
 .queryUsageAndQuota().
```

SPEICHERHANDLING

- requestFileSystem() fordert ein Speicherkontigent an, in der Regel ässt ein Browser 5 Mb pro Domain zu.
- Das Dateisystem ist durch die Sandboxing-Technologie geschützt ist, das heißt, dass eine Web-App nicht auf die Dateien einer anderen App zugreifen kann.
- Es kann keine Dateien lesen bzw. in einen beliebigen Ordner auf der Festplatte des Nutzers – beispielsweise "Eigene Bilder", "Eigene Dokumente" und so weiter – schreiben.

FILEENTRY/DIRECTORYENTRY

DAS FILEENTRY OBJEKT

- Es enthält Methoden und Eigenschaften für den Zugriff auf Dateieinträge:

- fileEntry.isFile === true
  fileEntry.isDirectory === false
  fileEntry.name
  fileEntry.fullPath
  ...
  fileEntry.getMetadata(successCallback, opt_errorCallback);
  fileEntry.remove(successCallback, opt_errorCallback);
  fileEntry.moveTo(dirEntry, opt_newName, opt_successCallback, opt_errorCallback);
  fileEntry.copyTo(dirEntry, opt_newName, opt_successCallback, opt_errorCallback);
  fileEntry.getParent(successCallback, opt_errorCallback);
  fileEntry.toURL(opt_mimeType);
  fileEntry.file(successCallback, opt_errorCallback);
  fileEntry.createWriter(successCallback, opt_errorCallback);
  ...

SO WIRD EINE DATEI I FILESYSTEM ANGELEGT

- Sie können im Dateisystem mithilfe von getFile(), einer Methode der DirectoryEntry-Schnittstelle, eine Datei suchen oder erstellen. Nach der Anforderung dieser Datei wird ein FileSystem-Objekt an das Erfolgs-Callback weitergegeben. Dieses Objekt enthält eineDirectoryEntry (fs.root)-Klasse, die auf das Stammverzeichnis des Dateisystems der App verweist.

- // Erstellen einer leeren Datei "log.txt" im Stammverzeichnis der App:

```
function onInitFs(fs) {
  fs.root.getFile('log.txt', {create: true, exclusive: true}, function(fileEntry) {
        // fileEntry.isFile === true
        // fileEntry.name == 'log.txt'
        // fileEntry.fullPath == '/log.txt'
  }, errorHandler);
}
window.requestFileSystem(
  window.TEMPORARY, 1024*1024, onInitFs, errorHandler
);
```

DATEIEN LESEN

- // Aufruf der Datei "log.txt" aufgerufen, Inhalt mit dem FileReader-API lesen
  // Inhalt in eine Textarea schicken, Fehlermeldung, falls Datei nicht existiert.

```
function onInitFs(fs) {
    fs.root.getFile('log.txt', {}, function(fileEntry) {
            // übernimmt ein File objekt, FileReader liest den Inhalt
            fileEntry.file(function(file) {
                    var reader = new FileReader();
                    reader.onloadend = function(e) {
                            var txtArea = document.createElement('textarea');
                            txtArea.value = this.result;
                            document.body.appendChild(txtArea);
                    };
                    reader.readAsText(file);
            }, errorHandler);
    }, errorHandler);
}
window.requestFileSystem(window.TEMPORARY, 1024*1024, onInitFs, errorHandler);
```

EINE DATEI SCHREIBEN

```
// Leere Datei "log.txt" erstellen, falls sie noch nicht existiert, Mit "Lorem ipsum" füllen.
function onInitFs(fs) {
    fs.root.getFile('log.txt', {create: true}, function(fileEntry) {
            // Ein FileWriter Objekt für den Dateieintrag "log.txt" erstellen.
            fileEntry.createWriter(function(fileWriter) {
                    fileWriter.onwriteend = function(e) {
                            console.log('Write completed.');
                    };
                    fileWriter.onerror = function(e) {
                            console.log('Write failed: ' + e.toString());
                    };
                    // Ein neues Blob erstellen und Inhalte schreiben.
                    var bb = new BlobBuilder(); // Note: window.WebKitBlobBuilder in Chrome 12.
                    bb.append('Lorem Ipsum');
                    fileWriter.write(bb.getBlob('text/plain'));
            }, errorHandler);
    }, errorHandler);
}
window.requestFileSystem(window.TEMPORARY, 1024*1024, onInitFs, errorHandler);
```

DATEN FORTSCHREIBEN

```
// Mit dem folgenden Code wird der Text "Hello World" an das Ende unserer
// Datei angehängt.
// Es wird ein Fehler ausgelöst, falls die Datei nicht vorhanden ist.

function onInitFs(fs) {
    fs.root.getFile('log.txt', {create: false}, function(fileEntry) {
            // Create a FileWriter object for our FileEntry (log.txt).
            fileEntry.createWriter(function(fileWriter) {
                fileWriter.seek(fileWriter.length); // Start write position at EOF.
                // Create a new Blob and write it to log.txt.
                var bb = new BlobBuilder();
                bb.append('Hello World');
                fileWriter.write(bb.getBlob('text/plain'));
            }, errorHandler);
    }, errorHandler);
}
window.requestFileSystem(window.TEMPORARY, 1024*1024, onInitFs,
errorHandler);
```

MEHRERE DATEIEN HOCHLADEN UND INS DATEISYSTEM DES BROWSERS KOPIEREN

- <input type="file" id="myfile" multiple />
```
document.querySelector('#myfile').onchange = function(e) {
    var files = this.files;
    window.requestFileSystem(window.TEMPORARY, 1024*1024, function(fs) {
            // Alle von Nutzer ausgewählten Dateien übertragen.
            for (var i = 0, file; file = files[i]; ++i) {
                    // Capture current iteration's file in local scope for the getFile() callback.
                    (function(f) {
                            fs.root.getFile(file.name, {create: true, exclusive: true},
  function(fileEntry) {
                            fileEntry.createWriter(function(fileWriter) {
                                    fileWriter.write(f); // Note: write() can take a File or Blob object.
                            }, errorHandler);
                            }, errorHandler);
                    })(file);
            }
    }, errorHandler);
};
```

DATEIEN ENTFERNEN

```
// Mit dem folgenden Code wird
// die Datei "log.txt" gelöscht.
window.requestFileSystem(
 window.TEMPORARY, 1024*1024, function(fs) {
    fs.root.getFile('log.txt', {create: false},
        function(fileEntry) {
            fileEntry.remove(function() {
            console.log('File removed.');
        }, errorHandler);
    }, errorHandler);
}, errorHandler);
```

DAS DIRENTRY OBJEKT MIT VERZEICNISSEN ARBEITEN

- // Eigenschaften und Methoden von DirectoryEntry:
  dirEntry.isDirectory === true
  // Vergleiche auch mit dem fileEntry Objekt.
  ...
  var dirReader = dirEntry.createReader();
  dirEntry.getFile(
    path, opt_flags, opt_successCallback, opt_errorCallback
  );
  dirEntry.getDirectory(
    path, opt_flags, opt_successCallback, opt_errorCallback
  );
  dirEntry.removeRecursively(
    successCallback, opt_errorCallback
  );
  ...

EIN VERZEICHNIS ERSTELLEN

- // Mit getDirectory() aus DirectoryEntry werden
  // Verzeichnisse gelesen oder erstellt.
  // Entweder Namen oder Pfad als zu suchendes
  // bzw. zu erstellendes Verzeichnis angeben.
  // Hier: ein Verzeichnis "MyPictures" erstellen:

```
window.requestFileSystem(
 window.TEMPORARY, 1024*1024, function(fs) {
      fs.root.getDirectory('MyPictures', {create: true},
      function(dirEntry) {
          ...
           }, errorHandler);
}, errorHandler);
```

DRAG&DROP

- Mit JavaScript-Bibliotheken wie jQuery ist es zwar möglich, Drag and Drop innerhalb eines Browsers zu realisieren, will man aber den Desktop einbeziehen oder mit verschiedenen Browsern arbeiten, kommt man an HTML5 nicht vorbei.

- Die dafür vorgesehene Drag-and-Drop-API wurde ursprünglich von Microsoft bereits 1999 mit dem IE5 auf den Markt gebracht, später von Safari und Mozilla übernommen (Chrome unterstützt sie ebenfalls) und fand deshalb Eingang in den HTML5-Standard.

- Eine saubere Cross-Browser-Verwendung macht aber, wie sollte es anders sein, ein wenig Mühe.

- Quelle: http://t3n.de/news/

TÜCKEN UND IMPLEMENTIERUNGSSCHWÄCHEN

- Alle Image- und Anker-Elemente sind per Default „draggable“, also für Drag-and-Drop verwendbar;

- Andere Elemente muss man mit dem HTML-Attribut draggable ausstatten;

- Webkit hält sich nicht an den Standard, hier muss der Draggable-Status über CSS gesetzt werden;

- im IE muss man, da er das draggable-Attribut nicht kennt, mit einem Trick nachhelfen und ein -Element, das draggable sein soll, zu einem <a>-Element umbauen;

- Das HTML-Attribut dropzone für die Zielzone des Drags wird von derzeit keinem Browser unterstützt.

EVENTS FÜR DAS DRAG AND DROP

Verschiedene DOM-Events stehen nun zur Verfügung, um mit dem Drag-and-Drop-Vorgang umzugehen,
zum Beispiel:

dragstart („Jetzt geht's los“),
drag („Es passiert!“)
dragged („Das war's, wir sind fertig“).

DATEN SCHICKEN UND EMPFANGEN

- `set Data(), getData()`
- Über setData lassen sich per Drag-and-Drop auch Daten übertragen, denen man einen MIME-Type mitgeben kann.
- Diese Daten können mit getData wieder ausgelesen werden.

EIN ELEMENT DRAGGABLE MACHEN

Img und Anchor Elemente sind von Haus aus draggable.
Andere Elemente erhalten das Attribut "draggable".

```
<div draggable="true">
    Drag me
</div>
```

WEBKIT BENÖTIGT CSS!

```
<div draggable="true">
    Drag me
</div>

[draggable="true"] {
  -khtml-user-drag: element;
  -webkit-user-drag: element;
  -khtml-user-select: none;
  -moz-user-select: none;
  -webkit-user-select: none;
  user-drag: element;
  user-select: none;
}
```

EIN ELEMENT DROPPABLE MACHEN

von Haus aus: Input, Textarea und alle Elemente, deren Inhalte bearbeitbar sind. Für alles andere gibt es ein "dropzone" Attribut.

```
<div dropzone="..."
     ondrop="handleDrop(event, this)">
    Drop here!
</div>
```

DROPPING

```
var dropzone = document.querySelector(".dropzone");
document.querySelector(".dragr").ondragstart = function(e)
{
    var dt = e.dataTransfer;
    dt.setData("text/plain", "I got dropped");
}
dropzone.ondrop = function(e) {
    var dt = e.dataTransfer,
    elem = this;
    elem.textContent = dt.getData("text/plain");
    e.preventDefault();
}
dropzone.ondragenter = function(e){e.preventDefault();}
dropzone.ondragover = function(e){e.preventDefault();}
```

DRAGGING AUF DEN DESKTOP (DERZEIT NUR CHROME)

```
<a href="assets/html5-cheat-sheet.pdf"
    class="button dragout"
    data-downloadurl="
         application/pdf:
         HTML5CheatSheet.pdf:
         http://url-der-datei/dateiname.pdf"
>Zieh mich auf den Desktop</a>
```

FORMATE, STEUERUNG, MÖGLICHKEITEN

AUDIO UND VIDEO MIT HTML5

DAS VIDEO - ELEMENT

- Oft wird das Video-Tag in den Medien als eine Flashalternative dargestellt, aber es ist mehr als das.

- Sein Hauptvorteil liegt in der natürlichen Integration in die anderen Ebenen der Web-Entwicklung, wie CSS und JavaScript und die anderen HTML-Elemente.

- Im Folgenden wird das Video-Tag und verschiedene Beispiele für unterschiedliche Integrationen in andere HTML5-Funktionen, beispielsweise <canvas> gezeigt.

DIE AUSZEICHNUNG <VIDEO>

- `<video src="movie.webm"></video>`

- Dies ist die einfachste Form, eine Videodatei in ein HML Dokument einzubinden. Die Syntax ähnelt dem Image_Element.

- Dies wird in naher Zukunft, wenn alle Browser ein gemeinsames Videoformat unterstützen, die meistgenutzte Syntax sein.

<VIDEO> UND <SOURCE>

```
<video>
  <source src="movie.mp4"
   type='video/mp4;   codecs="avc1.42E01E, mp4a.40.2"'
/>
  <source src="movie.webm"
   type='video/webm;
   codecs="vp8, vorbis"'
/>
</video>
```

In diesem Snippet wird das <source>-Tag verwendet, mit dem Sie mehrere Formate als Fallback-Typen aufnehmen können für den Fall, dass der Browser des Nutzers eines davon nicht unterstützt.

EINSTELLEN DES SERVERS FÜR VIDEOS

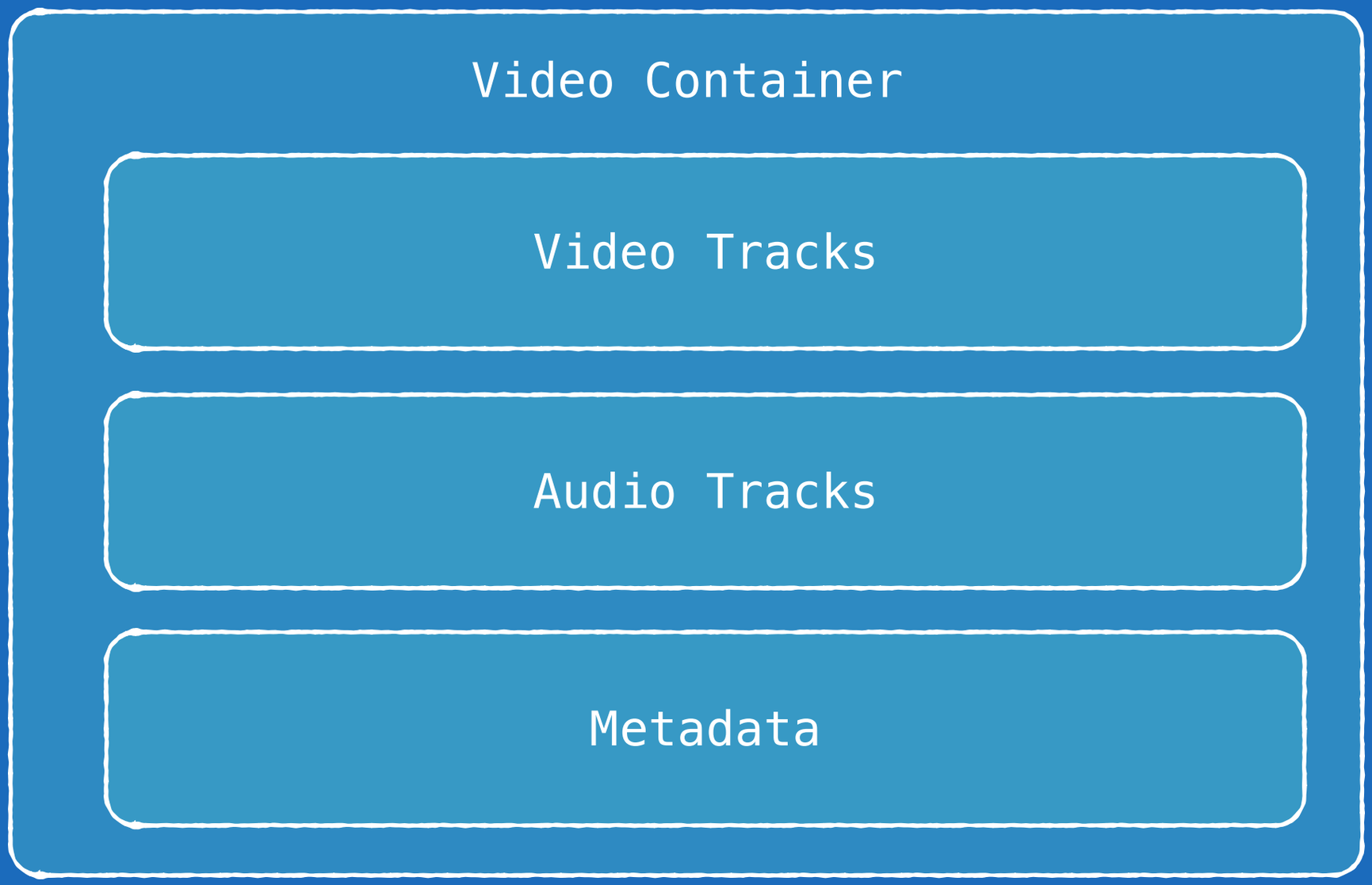
- Der Webserver muss Videodateien mit dem korrekten MIME-Typ im Content-Type-Header bereitstellen. Im anderen Fall funktionieren die Videos möglicherweise nicht ordnungsgemäß.

- In einer Apache-Datei "httpd.conf" werden beispielsweise diese Zeilen hinzugefügt:

- `AddType video/ogg .ogv`
- `AddType video/mp4 .mp4`
- `AddType video/webm .webm`

VIDEOCONTAINER

- Eine Audio- oder Videodatei ist eine Containerdatei, ähnlich einem Zip-Archiv, das mehrere Dateien enthält.



VIDEOCONTAINER

- Die Audio und Videospuren werden zur Laufzeit zusammengefügt.

- Die Metadaten enthalten Informationen über den Film, wie das Cover, den Titel, Untertitel und so weiter.

VIDEOFORMATE

- Die drei Formate, die für das Web gebraucht werden, sind "webm", "mp4" und "ogv":

- `.mp4` = H.264 + AAC
- `.ogg/.ogv` = Theora + Vorbis
- `.webm` = VP8 + Vorbis

- Diese sind als Standardformate definiert.
  Dies heisst aber nicht, dass Browser nicht auch andere Formate abspielen können.

AUDIO- UND VIDEOCODECS

- Codecs (coders/decoders) sind Algorithmen, mit denen Audio- und Videodateien zum Abspielen codiert bzw. decodiert werden.

- Sie komprimieren die Daten, so dass die zum Beispiel über das Netz übertragen werden können

- Ohne passenden Decoder kann der Empfänger die Datei nicht öffnen.

BROWSERUNTERSTÜTZUNG

- Heute unterstützen alle modernen Browser HTML5-Video, auch IE9.
- Für Firefox, Chrome und Opera wurde in Bezug auf Codecs vereinbart, über das WebM-Projekt zusätzlich ".webm" als gemeinsames Videoformat zu unterstützen.
- Internet Explorer unterstützt dies ebenfalls, sofern der Codec auf dem Computer installiert wird.

EINSCHRÄNKUNGEN

- Kein Streaming Audio oder Video, da im Moment kein Standard für Bitraten existiert. In Zukunft kann sich das ändern
- Es gelten die Regeln des Cross Site Managements.
- Vollbildvideo ist nicht programmierbar, da es gegen Sandbox-Sicherheitsrichtlinien verstößt. Über den Browser kann es aber eingestellt werden.
- Barrierefreiheit für Audio- und Videodaten ist noch nicht vollständig spezifiziert. Die Spezifikation "WebSRT" für Untertitelunterstützung auf Basis des SRT Formates ist in Arbeit.

PRÜFEN, OB <VIDEO> ZUR VERFÜGUNG STEHT

```
var hasVideo = (document.createElement('video').canPlayType);
```

Erzeugt dynamisch ein Video Element und prüft, ob die Methode canPlay vorhanden ist. So kann bei fehlender Unterstützung eine Fallbackposition, zum Beipsiel für ein Flashvideo eingebaut werden.

ALTERNATIVE AUSGABE

```
<video src="video.ogg" controls>
   Your browser does not support HTML5 video.
</video>
```

VIDEO FÜR ALLE BROWSER HEUTE

```
<video>
  <source src="movie.webm" type='video/webm; codecs="vp8, vorbis"' />
  <source src="movie.mp4" type='video/mp4; codecs="avc1.42E01E, mp4a.
40.2"' />
  <source src="movie.ogv" type='video/ogg; codecs="theora, vorbis"' />
  Video tag not supported.
  Download the video <a href="movie.webm">here</a>.
<video>
```

Hinweis: Aufgrund eines Fehlers im iPad
müssen Sie ".mp4" als erste Option angeben,
damit das Video auf diesem Gerät geladen wird.
Dies gilt, bis der Fehler behoben ist.

STANDARDVIDEO

Da sich fast alle Browseranbieter darauf geeinigt haben, ein gemeinsames Videoformat zu unterstützen, wird dieser Code im Web verwendet werden:

```
<video>
   <source   src="movie.webm"
             type='video/webm;
                   codecs="vp8, vorbis"' />
   <source   src="movie.mp4"
             type='video/mp4;
                   codecs="avc1.42E01E, mp4a.40.2"' />

   Video tag not supported.
   Download the video <a href="movie.webm">here</a>.
<video>
```

MP4 IST PROBLEMATISCH

- Zu den größten Bedenken bei der Verwendung des MP4-Formats zählt die Tatsache, dass sein Video-Codec (h.264) kein offener Codec ist und ein Unternehmen für seine Nutzung Lizenzen bezahlen müsste.

- Ein weiteres Problem mit dem MP4-Format besteht darin, dass das type-Attribut je nach dem für die Codierung des Videos verwendeten Profil spezifischer sein muss als bei anderen Formaten.

- Obwohl es in aller Regel "avc1.42E01E, mp4a.40.2" lautet, sollten Sie nochmals in dieser Profilliste nachsehen, um sich zu vergewissern.

DAS VIDEO ELEMENT IN ALTEN IE VERSIONEN?

- Das Google Chrome Frame Plugin

- versetzt Internet Explorer nach seiner Installation in die Lage, die neuesten HTML-, JavaScript- und CSS-Standardfunktionen zu unterstützen.

- Für Entwickler bietet dieses Plug-in den Zusatznutzen, dass sie damit Anwendungen mit modernen Webfunktionen codieren können, auf die auch die IE-Nutzer nicht verzichten müssen. Stellen Sie sich nur vor, wie viel Zeit ein Webentwickler einspart, wenn er keine Hacks und Behelfslösungen für IE codieren muss.

FLASH-FALLBACK

- Sie können auch Flash als Fallback verwenden. Je nach dem Format Ihres Videos müssen Sie dieses unter Umständen erneut codieren, damit es ein mit Flash Player kompatibles Format besitzt.

- Die gute Nachricht ist, dass Adobe sich dazu verpflichtet hat, das webm-Format in Flash Player zu unterstützen.

-
```
<video src="video.ogg">
 <object data="videoplayer.swf"
         type="application/x-shockwave-flash">
   <param name="movie" value="video.swf"/>
 </object>
</video>
```

<VIDEO> UND HTML

VIDEO UND ANDERES HTML

```
<video  poster="star.png"
       autoplay loop controls tabindex="0">
    <source src="movie.webm"
    type='video/webm; codecs="vp8, vorbis"' />
    <source src="movie.ogv"
    type='video/ogg; codecs="theora, vorbis"' />
</video>

Hier wird ein  tabindex verwendet, damit über
die Tastatur auf den Player zugegriffen werden kann.
```

ATTRIBUTE DES VIDEO - ELEMENTES

- `autoplay`
  für die automatische Wiedergabe eines Video- oder Audio - Elementes.

- `loop`
  Stellt die Mediendatei auf
  automatische Wiederholung ein.

- `currentTime`
  Gibt die aktuelle Zeit in Sekunden zurück, die seit dem Start des Films vergangen ist. CurrentTime kann auch gesetzt werden, um eine bestimmte Position im Film anzusteuern.

- `preload`
  Das Video wird im Hintergrund heruntergeladen, sobald die Seite geladen wird, selbst wenn seine Wiedergabe noch nicht begonnen hat.

WEITERE ATTRIBUTE

- `poster`
  Mit dem poster-Attribut wird angegeben, welches Bild angezeigt wird, wenn das Video am Anfang geladen wird.

- `controls`
  Der Browser soll automatisch Steuerelemente rendern. (Es werden keine benutzerspezifischen Steuerelemente erstellt.)

- `width, height`
  Read or set the visual display size. This may cause centering, letterboxing, or pillaring if the set width does not match the size of the video itself.

- `videoWidth, videoHeight`
  Return the intrinsic or natural width and height of the video. They cannot be set.

PROGRAMMIERBARE EIGENSCHAFTEN

- `volume`
  Stellt die Lautstärke des Films auf einen Wert zwischen 0.0 und 1.0. Der wert kann auch abgefragt werden.

- `muted`
  Stellt den Film auf stumm oder hörbar und liefert den aktuellen Status von mute zurück.

READ ONLY

- `duration`
  Liefert die volle Läge eines Clips in Sekunden. Fall der Wert unbekannt sein sollte, wird NaN ausgegeben.

- `paused`
  Liefert true, falls der Clip gerade pausiert. Ist per Default auf true gesetzt, wenn der Film noch nicht gestartet wurde.

- `ended`
  Liefert ein true, wenn der Film zu Ende gespielt hat.

- `startTime`
  Liefert die fühestmögliche Startzeit des Films. Das ist normalerweise ein 0.0 Wert, solange der Clip geladen wird und kein alter Content im Buffer liegt.

READ ONLY

- `error`
  Gbit einen Fehlercode aus, falls ein Fehler auftritt.

- `currentSrc`
  Gibt einen String zurück, der den Dateinamen enthält.

<VIDEO> UND JAVASCRIPT

VIDEO UND JAVASCRIPT

- `load()`
  Lädt eine Mediendatei und bereitet sie zum Abspielen vor. Der Befehl wird bei dynamischen geladenen Filmen angewandt. (Nicht bei im HTML stehenden).

- `play()`
  Lädt und spielt einen Film ab. Der Film wird entweder von der Startposition oder der aktuellen Pausenposition abgespielt.

- `pause()`
  Pausiert einen aktiven Film.

- `canPlayType(type)`
  Prüft, ob ein Browser über eine Playmethode verfügt oder nict. Damit kann die Unterstützung des Video - Elementes geprüft werden.

EVENTLISTENER FÜR DAS <VIDEO> ELEMENT

- Wie bei jedem anderen HTML-Element können Sie gängige Ereignisse an das Video-Tag anhängen, beispielsweise Ziehereignisse, Mausereignisse, Fokusereignisse und so weiter.

- Das Videoelement verfügt jedoch über eine Reihe von neuen Ereignissen, anhand derer erkannt und gesteuert wird, wann das Video abgespielt, angehalten oder beendet wird.

```
Events auf Netzwerkebene
loadstart
progress
suspend
abort
error
emptied
stalled

Events auf Zwischenspeicherungsebene
loadedmetadata
loadeddata
waiting
playing
canplay
canplaythrough
```

DER EVENTLISTENER CANPLAY

Auf einfachster Ebene können Sie als Einstieg für Ihre Beschäftigung mit dem Video das "canplay"-Ereignis anhängen.

```
video.addEventListener('canplay', function(e) {
    this.volume = 0.4;
    this.currentTime = 10;
    this.play();
}, false);
```

<VIDEO> UND CSS

VIDEO UND CSS

- Das <video> kann als Toplevel Element des DOM mit allen CSS Attributen ausgestattet werden.

- Dazu gehören Kanten, Deckkraft, aber auch CSS3-Eigenschaften wie Reflexionen, Masken, Farbverläufe, Transformationen, Übergänge und Animationen.

- ```
  #video-css.enhanced {
   border: 1px solid white;
   box-shadow: 0px 0px 4px #ffffff;
   transform: translate(0, 10px);
  }
  ```
  ```

GEOLOCATION API

GEOLOCATION API

- Mit Hilfe der Geolocation API kann der User den seinen geografischen Standort ermitteln lassen. (Um ihn zum Beispiel anderen Usern mitzuteilen).

- Und, wenn er einverstanden ist, kann die Webanwendung ihm dann einen Tipp geben, wo es in der Gegend ein günstiges Angebot für Schuhe gibt.

- Eine andere Idee wäre eine turn-by-turn GPS-style Navigation, oder eine social networking application, die anzeigt, wo sich die Freunde gerade aufhalten.

LÄNGEN- UND BREITENGRADE

- Die Ortsinformation besteht im wesentlichen in der Angabe von Längen- und Breitengrad der aktuellen Position, wie das folgende Beispiel zeigt. Es sind die geographischen Angabe für die Stadt Stuttgart:

- `geographische Breite: 48,755`
  `geographische Länge: –9.175`

- Der nördliche Abstand von Äquator beträgt also 48,755 Grad, die östliche Länge 9,175 Grad ausgehend von Greenwich Meridian, der den Nullmeridian liefert.

- Normalerweise werden die geographischen Angabe in Grad, Minuten und Sekunden angegeben:

- -9° 10' 30" bzw. 48° 47' 30"

WEITERE ANGABEN

- `accuracy, altitude, altitudeAccuracy, heading, speed.`

- Zusätzlich ermittelt die Geolocation API die Genauigkeit der ermittelten Angaben. Ausserdem können je nach Gerät die Höhenangabe, die Genauigkeit der Höhenangabe, die Richtung und die Geschwindigkeit ermittelt werden.

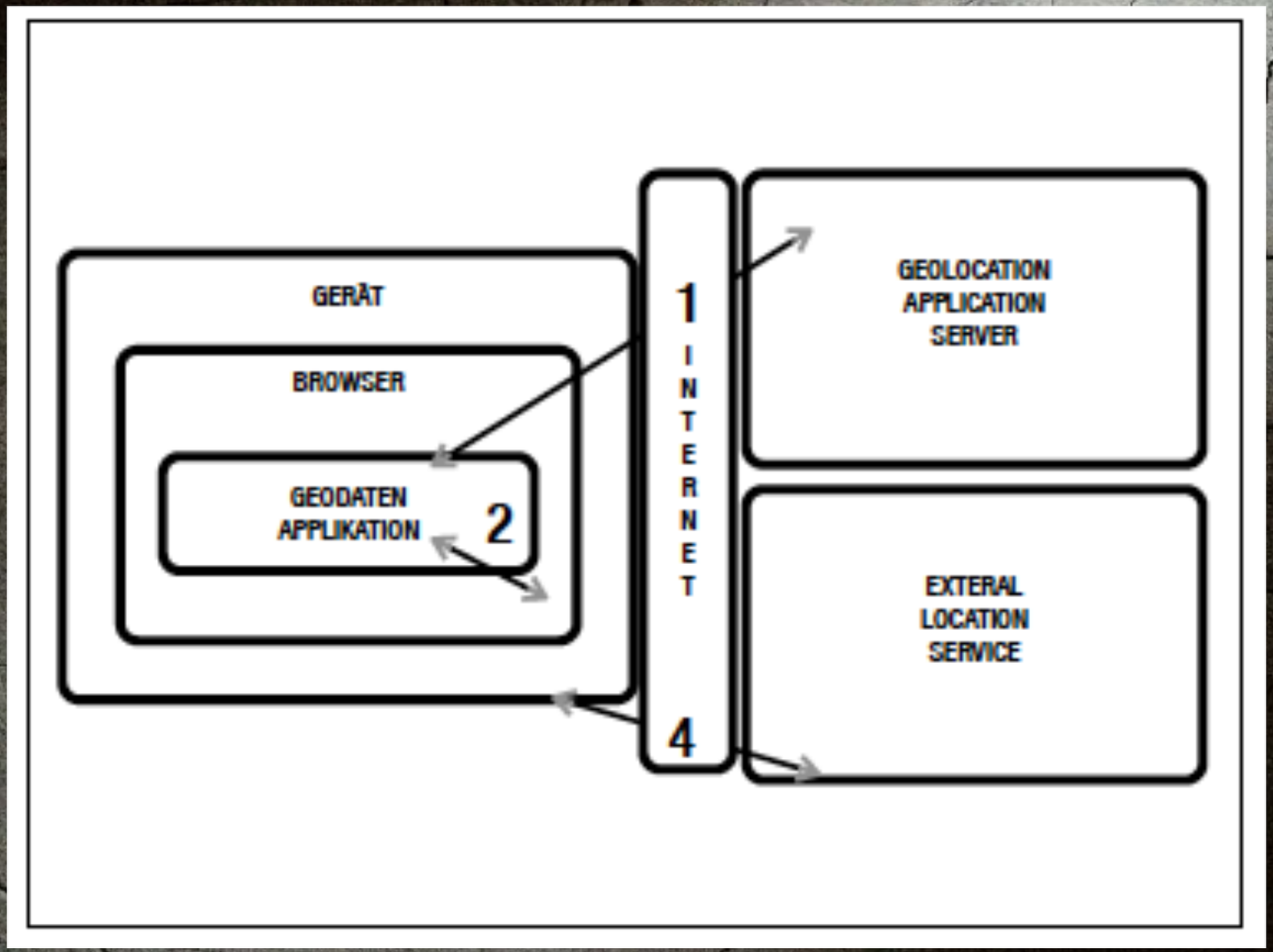
- Wenn keine Werte vorliegen wird jeweils Null zurückgegeben.

- Ein Gerät kann auf folgende Resourcen zur Koordinatenermittlung zurückgreifen:
- IP Adresse
- Koordinatentriangulation (Mittelwertberechnung)
- Global Positioning System (GPS)
- Wi-Fi mit MAC Adressen aus RFID-, Wi-Fi oder
- Bluetoothquellen
- GSM oder CDMA Mobliephone IDs
- Usereingaben
- Viele Geräte verwenden eine Kombination mehrerer Quellen, um eine höhere Genauigkeit im Ergebnis zu erzielen.

SCHUTZ DER PRIVATSPHÄRE

- Die HTML5 Geolocation Spezifikation fordert einen Mechanismus zum Schutz der Privatspäre des Users. So muss der User den Zugriff des Browsers auf seine Geodaten erlauben.

- Allerdings ist offen, was nach der Genehmigung mit den Daten geschieht, oder u.U. auch, wie lange sie gilt.


GERÄT
BROWSER
GEODATEN
APPLIKATION
2
1
INTERNET
4
GEOLOCATION
APPLICATION
SERVER
EXTERAL
LOCATION
SERVICE

- Oft ist es ausreichend, die Position nur einmal abzufragen.
- `navigator.geolocation.getCurrentPosition(updateLocation, handleLocationError);`
- Man kann sie auch permanent aktualisieren
- `navigator.geolocation.watchPosition(updateLocation, handleLocationError);`

GEOLOCATION PRÜFEN

```
function loadDemo() {
   if(navigator.geolocation) {
      document.getElementById("support").innerHTML
      = "HTML5 Geolocation supported.";
   } else {
      document.getElementById("support").innerHTML
      = "HTML5 Geolocation is not supported.";
   }
}
```

FEHLERBEHANDLUNG

```
function handleLocationError(error) {
    switch(error.code){
        case 0: // UNKNOWN ERROR
            updateStatus("There was an error while
            retrieving your location: " +
            error.message);
            break;
    case 1: // PERMISSION DENIED
            updateStatus("The user prevented this
            page from retrieving a location.");
            break;
```

```
        case 2: // POSITION_UNAVAILABLE
        updateStatus("The browser was unable to
        determine your location: " +
        error.message);
              break;
    case 3: // TIMEOUT
              updateStatus("The browser timed out
              before retrieving the location.");
              break;
        }
    }
```