Internetanwendungen für mobile Geräte – Übungsprogramm

Virtuelle Fachhochschule, Wintersemester 2015/2016, Stand: 2. September 2015

Jörn Kreutel

Inhaltsverzeichnis

Hinweise zur Bearbeitung

Bitte beachten Sie, dass die Übungen ausschließlich unter Verwendung der durch Firefox unterstützten Ausdrucksmittel von HTML, CSS und JavaScript umzusetzen sind. Die Verwendung externer Bibliotheken wie jQuery ist nicht zulässig, es sei denn es wird ausdrücklich darauf hingewiesen.

Die Abgabe der Aufgaben erfolgt in den folgenden 2-3 Blöcken zu Terminen, die konkret im Laufe des Semesters unter Berücksichtigung des Lernfortschritts festgelegt werden:

- Abgabe 1: Aufgaben zu HTM, CSS, JSL, JSR
- Abgabe 2a: Aufgaben zu NJM, FRM, MFM
- Abgabe 2b: Aufgaben zu MME, LDS, OFF

Die Aufgaben von Abgabe 2a und Abgabe 2b können ggf. in einer Abgabe zusammengefasst werden. Die Funktionalität der Aufgaben pro Abgabe baut jeweils in der genannten Reihenfolge aufeinander auf. Pro Abgabe sind daher alle Aufgaben jeweils in *einer* integrierten Anwendung zusammenzufassen, die alle Anforderungen aller Aufgaben der Abgabe erfüllt. Beispielsweise muss die Umsetzung der Aufgaben zu JSR die Anforderungen zu (HTM), CSS und JSL erfüllen. Dafür müssen jeweils gemeinsame HTML, CSS oder JavaScript Ressourcen genutzt werden. Selbiges gilt für die Aufgabenblöcke 2a und 2b insgesamt, d.h. die für Abgabe 2a umgesetze Funktionalität muss in die Umsetzung der Aufgaben für Abgabe 2b unter Verwendung gemeinsamer Ressourcen integriert sein.

Bezüglich der Umsetzung der Aufgaben können Sie entscheiden, ob Sie die bereit gestellten Implementierungsbeispiele verwenden und deren Architektur übernehmen oder ob Sie die Anwendung unabhängig davon entsprechend Ihren eigenen architektonischen Vorstellungen entwickeln.

Die in grau wiedergegebenen Aufgaben dienen der Durchführung vorbereitender Maßnahmen bzw. der Wiederholung ausgewählter Aspekte des Lernstoffs anhand der Implementierungsbeispiele. Für die Abgaben sind diese Aufgaben nicht relevant.

Für die Bearbeitung der Aufgaben ist es empfehlenswert, dass Sie zunächst die gesamte Aufgabenbeschreibung, inklusive der Bearbeitungshinweise, durchlesen und erst dann mit der Umsetzung beginnen.

HTM

Ü HTM1 Inbetriebnahme der Beispiele

1. Laden Sie die Implementierungsbeispiele aus dem Moodle herunter.
2. Öffnen Sie Aptana und importieren Sie die Beispiele (*Import...* → *General* → *Existing Projects into Workspace* → *Select Archive File*) in Ihren Workspace.
3. Überprüfen Sie die Server-Einstellungen in Aptana (*Einstellungen* → *Aptana Studie* → *Web Servers* → *Built-in*) und wählen Sie als IP-Adresse `127.0.0.1` aus. Starten Sie Aptana danach neu.
4. Greifen Sie auf die URL `http://127.0.0.1:8020/org.dieschnittstelle.iam.htm/index.html` zu bzw. führen Sie auf `index.html` die Aktion *Run As...* → *JavaScript Web Application* aus.
5. Sie sollten nun auch auf alle Beispiele, mit denen die Präsentation verknüpft ist, zugreifen können.

Sonstiges

Demovideo: `https://connect.oncampus.de/p43gjzffmei/`

Ü HTM2 Inspizieren der Beispiele

1. Öffnen Sie Firebug.
2. Wählen Sie den Tab `Net` aus und aktivieren Sie ggf. die Verfolgung des Netzwerkverkehrs.
3. Rufen Sie ausgehend von `index.html` die einzelnen Beispiele auf.
4. Im `Net` Tab sehen Sie, auf welche Ressourcen für die einzelnen Beispiele zusätzlich zum aufgerufenen Dokument zugegriffen wird.
5. Die Beispiele dienen nur dem zusammenfassenden Überblick 'von außen' über die Ausdrucksmittel von HTML5 und nicht der Vermittlung von Implementierungskompetenzen – im einzelnen werden wir uns damit (auszugsweise) im Verlauf des Semesters beschäftigen.

Sonstiges

Demovideo: `https://connect.oncampus.de/p3fct7r5bz4/`

Ü HTM3 Dokumentenstruktur

Aufgabe

Konzipieren Sie eine geeignete Dokumentenstruktur für die nachfolgende Ansicht unter Berücksichtigung des in der Lerneinheit vermittelten Stoffs und unter Ausschöpfung der Ausdrucksmittel von HTML5. Beachten Sie bitte, dass dieser Aufgabe lediglich vorbereitender Charakter für die folgenden Aufgaben zukommt. Daher werden hier auch keine Punkte für die Bewertung angegeben. Konkret umsetzen müssen Sie für die abzugebenden Aufgaben nur Struktur und Styling der Gestaltungselemente 'Medienverweise', 'Textauszug', sowie 'Imgbox'. Letzteres Element wird in Ü CSS3 beschrieben.

Bearbeitungshinweise

- Das Element *Auf meine Merkliste* brauchen Sie nicht zu berücksichtigen.
- Bedenken Sie insbesondere die folgenden Punkte:
  - Welche Elemente verwenden Sie für die Kopf- und Fußleiste?
  - Wie repräsentieren Sie die Inhalte der Fußleiste? Bedenken Sie, dass es sich hierbei um eine Menge inhaltlich gleichgeordneter Optionen handelt. Welches (bereits in der Urfassung existierende) HTML Element ist hierfür geeignet?
  - Alle Gestaltungselemente der Ansicht stellen jeweils verschiedenes Material zu einem gemeinsamen Thema ('Die Umsiedlerin') zur Verfügung. Welche Elemente von HTML5 eignen sich hierfür?
  - Angenommen, alle Elemente, von denen in der Ansicht nur ein Exemplar existiert (*'Zeitdokumente'*, *'Bilder'*, *'Kommentar'* (blau), *'Textauszug'*, *'Mehr unter'* können nicht mehrfach auftreten: Wie können Sie diese Elemente identifizierbar machen?
  - Wie kann die Tatsache, dass ein Elementtyp (*'Medienverweise'*) in zwei Exemplaren auftritt ('Kommentar', 'Filme und Audio') zum Ausdruck gebracht werden und gleichzeitig die eindeutige Identifizierbarkeit der Exemplare gewährleistet werden?
  - Welche HTML5 Standardelemente bieten sich für die Ansicht *'Zeitdokumente'* an?
  - Wie realisieren Sie die Aktionen `weiter lesen` in *'Kommentar'* und *'Textauszug'*?

- Wie könnte der Text in *'Textauszug'* möglichst fein granular strukturiert werden? Schauen Sie sich dafür noch einmal die Ausführungen im Skript an. Wie könnte man den dargestellten Inhalt noch weiter strukturieren?
- Wie bauen Sie die Elemente für 'Medienverweise' auf?

CSS

Ü CSS1 Entwicklungswerkzeuge

- Ein CSS Editor mit Codevervollständigung steht Ihnen in Aptana zur Verfügung.
- Als 'offizielle' Validierungsinstanz können Sie auch für CSS einen Service des W3C nutzen: `http://jigsaw.w3.org/css-validator/#validate_by_upload`
- Firefox bietet Ihnen die Kontextmenü-Aktion *Inspect Element* an, auf die hin der *Firefox Inspector* geöffnet wird. Dieser verfügt u.a. über die folgende Funktionalität:
  - Visualisierung des aktuellen Zustands des HTML Dokuments, inklusive aller durch JavaScript vorgenommenen Manipulationen (siehe auch die folgende Lerneinheit), mit WYSIWYG Editierfunktion.
  - Auflistung der zur Darstellung des ausgewählten Elements verwendeten CSS Regeln mit Visualisierung überschriebener Property-Setzungen und WYSIWYG Editierfunktionen (Achtung: Änderungen müssen Sie dann ggf. manuell in die CSS Originaldatei übernehmen).
  - Darstellung der Realisierung des Box Model für das ausgewählte Element.
  - weitere Dokumentation finden Sie hier: `https://developer.mozilla.org/en-US/docs/Tools/Page_Inspector`.
- Zum Testen verschiedener Bildschirmgrößen können Sie in Firefox die Aktion *Extras* → *Web-Entwickler* → *Bildschirmgrößen testen* verwenden.

Ü CSS2 Inspizieren der Beispiele

- Importieren Sie die Implementierungsbeispiele in Ihren Workspace.
- 'Starten' Sie die Datei `css.html` in Aptana (d.h. öffnen Sie die URL `http://127.0.0.1:8020/org.dieschnittstelle.iam.css/css.html`
- Vergleichen Sie die gewählte HTML Struktur mit der von Ihnen in Ü HTM umgesetzten. Worin bestehen die wesentlichen Unterschiede?
- Beschäftigen Sie sich mit der Verwendung von CSS zur Darstellung der nachfolgend genannten Elemente.

'Zeitdokumente'

- Wie verändert sich die Darstellung, wenn Sie als Bildquelle `http://lorempixel.com/400/250` zuweisen?
- Welche Attribute sind für diese Änderung verantwortlich?
- Ist die Zuweisung des Schrifttyps in der Regel für `#zeitdokumente` erforderlich?

'Einführungstext'

- Warum ist die explizite Zuweisung von `font-family` hier erforderlich?
- Wir wird das 'Abschneiden' des Texts umgesetzt?
- Wie erfolgt die Positionierung des Pfeils vor 'weiter lesen'?

'Verknüpfungen' ('Mehr dazu...')

- Welche Strukturelemente werden hierfür verwendet? Sind anwendungsspezifische Elemente (`div`, `span`) erforderlich?
- Welche Elemente verwenden die gestrichelte Linie als Umrandungs- bzw. Trennlinie?
- Wie wird die Bündigkeit von Trennlinie und der Bezeichnung der Optionen 'Ort' und 'Zeit' umgesetzt, und wie der gemeinsame Abstand zur Umrandung?
- Welche Property ist für die Großschrift der Überschrift verantwortlich?
- Wie wird das Unterstreichen der Links in den `<a>`-Elementen unterbunden?

Sonstiges

Demovideo: `https://connect.oncampus.de/p2maexobt9c/`

Ü CSS3 Einzelne Gestaltungselemente (21 Punkte)

Aufgabe

Setzen Sie die Elemente ‘Medienverweise’, ‘Textauszug’, ‘Imgbox’ entsprechend den nachfolgend skizzierten und umseitig abgebildeten Gestaltungsvorgaben um.

Anforderungen

1. Medienverweise
   (a) Überschrift ***(1 P.)***
   (b) Titelliste ***(1 P.)***
   (c) Hintergrund ***(1 P.)***
   (d) Trennlinien mit Rand ***(2 P.)***
   (e) Keine Trennlinie unter dem letzten Titel ***(1 P.)***
   (f) Pfeil ***(2 P.)***

2. Textauszug
   (a) Schneiden Sie unter Verwendung von CSS längere Texte nach unten ab ***(2 P.)***
   (b) Verwenden Sie anwendungsspezifisches ‘semantisches Markup’ für die Textstruktur ***(3 P.)***
   (c) Titel ***(1 P.)***
   (d) Text ***(1 P.)***
   (e) Verknüpfung mit Pfeil ***(2 P.)***

3. Imgbox
   (a) Titel ***(1 P.)***
   (b) Strichellinie ***(1 P.)***
   (c) Kringel ***(1 P.)***
   (d) Bild ***(1 P.)***

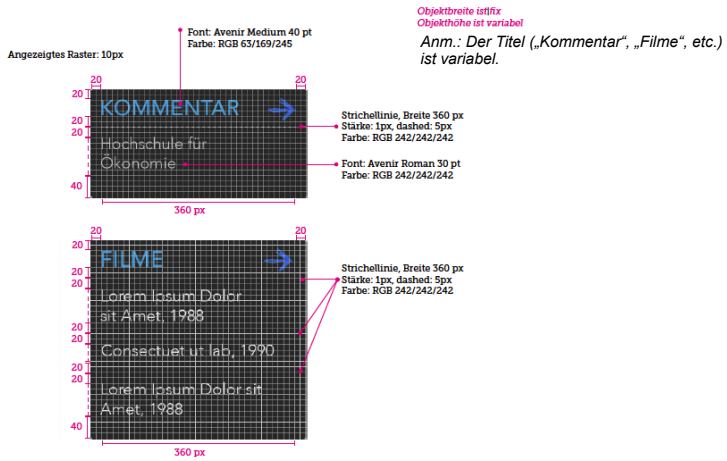
Bearbeitungshinweise

- Das Projekt `org.dieschnittstelle.iam.css.uebungen` können Sie als Ausgangspunkt für diese Aufgabe verwenden. In der folgenden Lerneinheit werden Sie dieses Projekt erweitern.
- Ändern Sie für die Umsetzung ggf. die von Ihnen in Ü HTM3 vorgesehene Dokumentstruktur.
- ***Anforderung 2***: ein Beispiel für das geforderte ‘Abschneiden’ finden Sie in der Umsetzung des Elements ‘Einführungstext’ in den aktuellen Beispielen.
- ***Anforderung 3***: verwenden Sie in ‘Imgbox’ ein Bild Ihrer Wahl, ggf. auch die in den Implementierungsbeispielen genutzten Bilder.
- Halbieren Sie die Größenangaben aus den Gestaltungsvorgaben (diese sind für ein ‘Retina’-Display angenommen) und nehmen Sie ggf. weitere Änderungen nach Augenmaß vor.
- Achtung: die Raster dienen der Angabe von Maßen und sind nicht umzusetzen. Schattierungen sind gleichermaßen zu vernachlässigen.
- Bildmaterial zur Realisierung von Pfeilen, Kringeln und Kreuzen finden Sie im Verzeichnis `css/img` des Projekts `org.dieschnittstelle.iam.css.uebungen`.
- Demovideo: `https://connect.oncampus.de/p4705x5azq9/` (alle Aufgaben zu CSS+JSL)

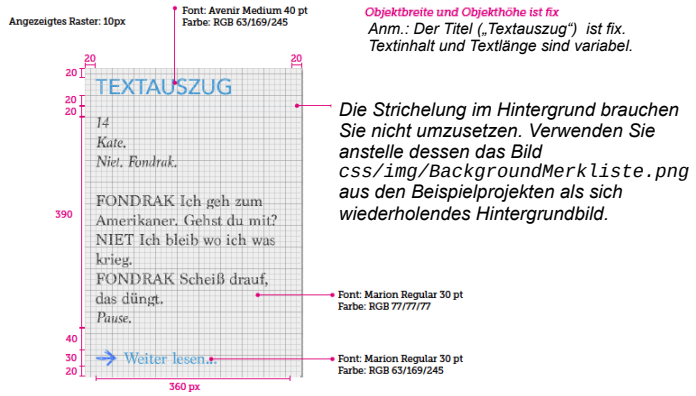
Sonstiges

- Vorlage ‘Medienverweise’

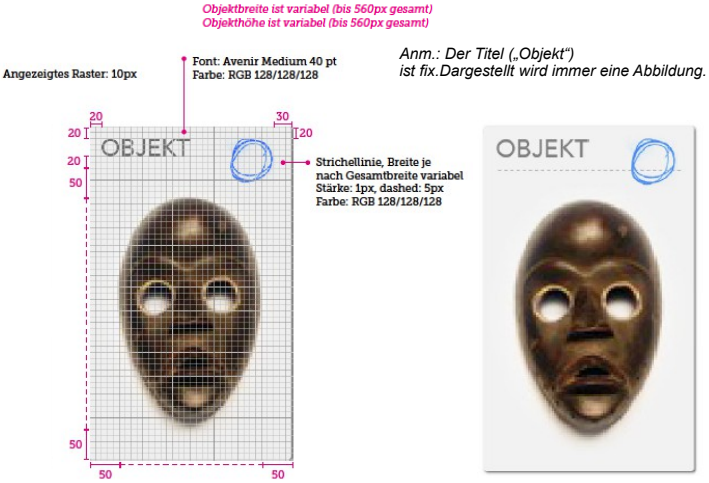
die Hintergrundfarbe ist RGB 45/45/45



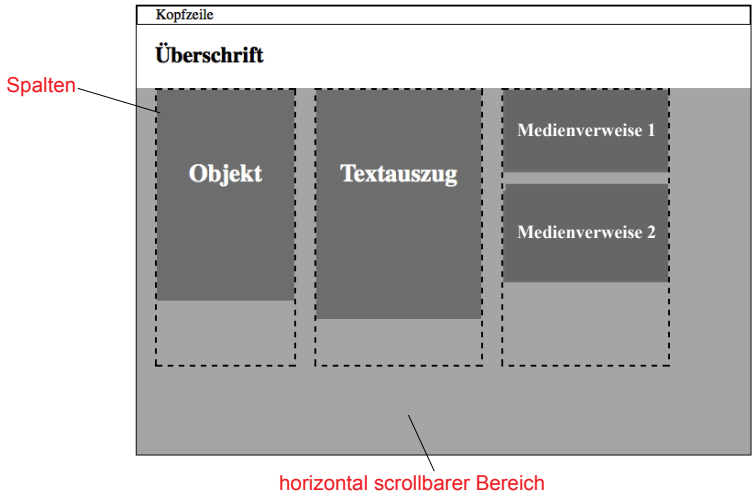
- Vorlage ‘Textauszug’:



- Vorlage ‘Imgbox’ (*nicht zu verwechseln mit dem Element ‘Zeitdokumente’*):



Ü CSS4 Realisierung des Gesamtlayouts (9 Punkte)



Aufgabe

Stellen Sie die drei in Ü CSS3 bearbeiteten Elemente, wie oben gezeigt, nebeneinander in einem Spaltenlayout dar.

Anforderungen

1. Vom Elementtyp 'Medienverweise' sollen wie in der Abbildung zu Ü HTM3 zwei Exemplare verwendet werden. ***(1 P.)***
2. Die Elemente sollen, wie in der Gestaltungsvorlage vorgesehen, nach oben bündig abschließen. ***(1 P.)***
3. Die linksaußen platzierte Spalte soll bündig mit der Überschrift ('Die Umsiedlerin') abschließen. ***(1 P.)***
4. Der Abstand zwischen den Spalten soll 20px betragen. ***(1 P.)***
5. Verhindern Sie, dass bei Verkleinerung der Breite des Browserfensters die Anordnung der Spalten geändert wird. ***(2 P.)***
6. Ermöglichen Sie, dass bei Verkleinerung der Breite des Browserfensters die Ansicht der Spalten insgesamt horizontal scrollbar ist ***(2 P.)***, ohne dass der Titel der Ansicht ('Die Umsiedlerin') und die Kopfzeile beim Scrollen verschoben werden ***(1 P.)***.

Bearbeitungshinweise

- ***Anforderung 5***: dafür können Sie z.B. dem Mutter-Element der Spalten eine fixe Breite zuweisen. Anhaltspunkte hierfür finden Sie auch in den Implementierungsbeispielen zu den JSL Übungen.
- ***Anforderung 6***: Ziehen Sie dafür z.B. in Betracht, ein zusätzliches `div` Element einzuführen, das mit der `overflow` Property für horizontales Scrolling versehen wird.

- ***Anforderung 6***: Der Punkt bezüglich der Fixierung von Kopfzeile und Überschrift wird nur vergeben, wenn horizontales Scrolling wie beschrieben möglich ist.

JSL

Ü JSL1 Implementierungsbeispiele

Spaltenlayout

- Wie ist die Nebeneinander-Anordnung der beiden `<section>` Elemente umgesetzt?
- Welche Rolle spielt dabei die Breitenzuweisung an das Element mit ID `articlecontent`?
  - Reduzieren Sie dafür die Breite z.B. auf 700px und schauen Sie sich den Unterschied an.
- Berücksichtigen Sie außerdem den Hinweis zur Dimensionierung der Eltern-Elemente von `float` Elementen *hier...*
- Als Alternative zur Verwendung von `float` könnten Sie ein `inline-block` Element in Verbindung mit einer geeigneten `vertical-align` Property verwenden.

Scrolling

- Verkleinern Sie die Breite des Browserfensters so, dass nur noch ein Ausschnitt der Ansicht zu sehen ist.
- Ihnen sollte jetzt die Möglichkeit gegeben sein, die Ansicht horizontal zu scrollen.
- Durch welche Style-Property auf `<article>` wird dies erreicht?

Positionierung von Elementen

- Wodurch wird die Positionierung des Toasts im unteren Bereich des Browserfensters herbeigeführt?

DOM Zugriff

- Einen Überblick über die Ausdrucksmittel zur DOM Manipulation erhalten Sie anhand einer Betrachtung aller Methoden im `demo.js` Skript und einer Vergegenwärtigung von deren Funktionalität anhand Ihrer Interaktion mit der Beispielanwendung.

Einblenden/Ausblenden

- Wie wird das Einblenden des ‘Toasts’ bzw. das Ausblenden der `<section>` Elemente in JavaScript veranlasst?
- Was ist erforderlich, um den ‘Toast’ nach einer gewissen Verweildauer wieder ausblenden zu können?

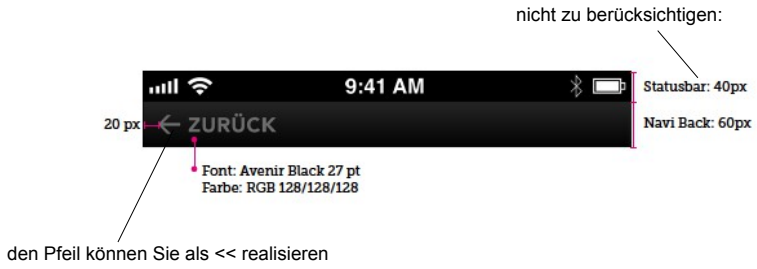
Sonstiges

Demovideo: `https://connect.oncampus.de/p879v50tsx3/`

Ü JSL2 Umsetzung der Kopfzeile (4 Punkte)

Aufgabe

Setzen Sie die Kopfzeile wie nachfolgend dargestellt um.



Anforderungen

1. Merkmale:
   (a) Fixe Höhe ***(1 P.)***
   (b) Hintergrund ***(1 P.)***
   (c) Beschriftung ***(1 P.)***
   (d) Die Kopfzeile soll ohne Zwischenraum bündig nach oben abschließen ***(1 P.)***

Bearbeitungshinweise

- Wählen Sie zur Realisierung des 'Zurück' Bedienelements ein geeignetes HTML Element, z.B. `<a>` oder `<button>`.
- Demovideo: `https://connect.oncampus.de/p4705x5azq9/` (alle Aufgaben zu CSS+JSL)

Ü JSL3 Umsetzung der Detailansicht (7 Punkte)

Aufgabe

Setzen Sie die nachfolgend beschriebene Detailansicht für Ihr Gestaltungselement ‘Textauszug’ so um, dass Gesamtansicht und Detailansicht dieselbe Dokumentstruktur verwenden und sich nur hinsichtlich der zugewiesenen Attribute unterscheiden. Orientieren Sie sich hinsichtlich der visuellen Gestaltung an den nachfolgend gezeigten Gestaltungsvorlagen.

Anforderungen

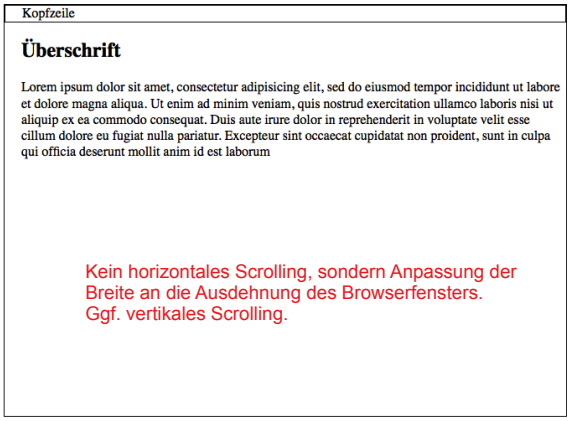
1. Merkmale:
   (a) Überschrift wird aus der Gesamtansicht übernommen (z.B. ‘Die Umsiedlerin’), die ‘Textauszug’ Überschrift ist nicht sichtbar. ***(1 P.)***
   (b) Hintergrund komplett weiß, kein Grau sichtbar. ***(1 P.)***
   (c) Der gesamte Text des ‘Textauszug’ Elements soll dargestellt werden.***(1 P.)***
   (d) Kein horizontales Scrolling, ggf. vertikales Scrolling. ***(1 P.)***
   (e) Text soll an die Fensterbreite angepasst werden, d.h. die gesamte Breite mit Abständen entsprechend der Gestaltungsvorlage soll ausgenutzt und der Text soll nicht abgeschnitten werden. ***(1 P.)***
   (f) Kopfzeile und Überschrift sollen bei vertikalem Scrolling fix bleiben. Oberhalb oder seitlich von Kopfzeile und Überschrift soll beim Scrollen kein Text sichtbar sein.***(2 P.)***

Bearbeitungshinweise

- Der Punkt bezüglich der Fixierung von Kopfzeile und Überschrift wird nur vergeben, wenn vertikales Scrolling wie beschrieben möglich ist.
- Verwenden Sie als Grundlage das in den zurückliegenden Übungen verwendete HTML Dokument, welches auch die anderen von Ihnen umzusetzenden Gestaltungselemente enthält.
- Versuchen Sie durch geeignete Verbindung von `class` und `id` Attributen sowie CSS Regeln alle anderen Elemente auszublenden und nur ‘Textauszug’ als Detailansicht darzustellen. Je nach gewählter HTML Struktur ist ggf. auch die Verwendung zusätzlicher `div` Elemente erforderlich.
- Wenn Sie die Attribute wieder entfernen / modifizieren, sollte ohne weitere Änderung am CSS wieder die Gesamtansicht angezeigt werden.
- Nachfolgend finden Sie zur weiteren Erläuterung einen schematischen Aufbau der Detailansicht, die Gestaltungsvorlage sowie ein Umsetzungsbeispiel.

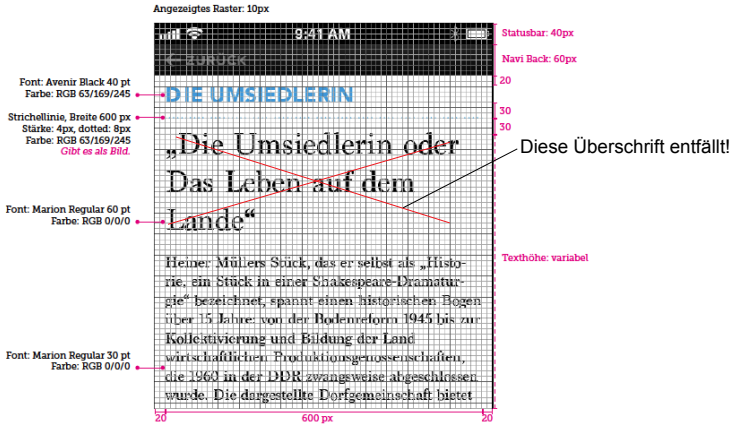
Sonstiges

- Schematischer Aufbau:

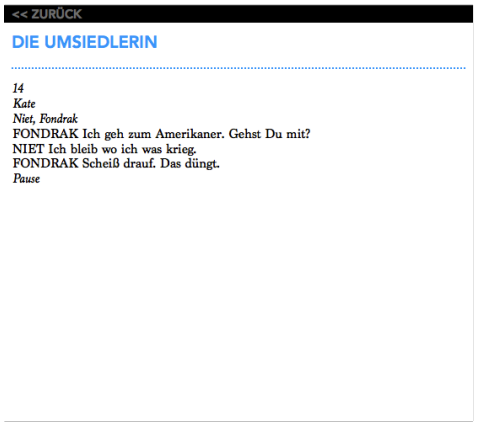


- Gestaltungsvorlage:

Setzen Sie entsprechend dieser Vorlage die Detailansicht zu „Textauszug“ um



- Umsetzungsbeispiel:



Ü JSL4 Ansichtsübergang (5 Punkte)

Aufgabe

Implementieren Sie in JavaScript den Übergang zwischen der Gesamtansicht und der von Ihnen in Ü JSL3 entwickelten Detailansicht sowie den Übergang zurück in die Gesamtansicht.

Anforderungen

1. Die Übergänge sollen *ohne Neuladen eines HTML Dokuments*, *ohne Entfernung von Elementen* aus dem DOM Objekt und *ohne Verwendung redundanter Inhalte* umgesetzt werden. ***(3 P.)***
2. Der Übergang in die Detailansicht wird durch Betätigung von 'Weiter lesen' in 'Textauszug' ausgelöst, die Rückkehr in die Gesamtansicht durch Betätigung von 'Zurück' auf der in Ü JSL2 umgesetzten Kopfzeile. ***(2 P.)***

Bearbeitungshinweise

- Ihr Javascript sollte die in Ü JSL3 manuell erstellten Zustände Ihres Dokuments für die Detailansicht und die Gesamtansicht durch geeignete Verwendung der DOM API herbeiführen.
- ***Anforderung 1***: redundant wäre z.B. die Verwendung zweier Elemente mit Texten unterschiedlicher Länge, um das 'Abschneiden' des Textauszugs in der Gesamtansicht gegenüber dem vollständigen Text in der Detailansicht zu realisieren.

Ü JSL5 Aus- und Einblenden (5 Punkte)

Aufgabe

Bauen Sie ein ‘Ausblenden’/‘Einblenden’ der Ansichten in den in Ü JSL4 umgesetzten Ansichtsübergang ein.

Anforderungen

1. Nach Auslösung eines Ansichtsübergangs durch den Nutzer soll zunächst die aktuelle Ansicht 2 Sekunden lang ausgeblendet werden. Danach soll die neue Ansicht 2 Sekunden lang eingeblendet werden. ***(3 P.)***
2. Die Kopfzeile wird nicht durch das Ausblenden/Einblenden beeinflusst, d.h. sie soll dauerhaft unverändert sichtbar sein. ***(2 P.)***

Bearbeitungshinweise

- Für das Aus/Einblenden können Sie die Style-Property `opacity` verwenden.
  - `opacity: 0.0`: ‘vollkommen transparent’, d.h. unsichtbar
  - `opacity: 1.0`: ‘vollkommen intransparent’, d.h. ohne durchschimmernden Hintergrund sichtbar.
- Ziehen Sie z.B. in Betracht, die Änderung der `opacity`-Property für das gemeinsame Wurzelelement der Ansicht unterhalb der Kopfzeile umzusetzen, falls ein solches in Ihrer HTML Struktur existiert.
- Den Ansichtswechsel nach Ausblenden der Gesamtansicht können Sie z.B. durch Setzen eines `transitionend` Event Listeners veranlassen. Eine Vorlage dafür finden Sie in der Umsetzung des ‘Toasts’ in den Implementierungsbeispielen.

JSR

Ü JSR1 Implementierungsbeispiele

XMLHttpRequest

- Vergegenwärtigen Sie sich die Funktionalität von `XMLHttpRequest` anhand der Implementierung der `xhr()` Funktion in der JavaScript-Datei `xhr.js` und ihrer Nutzung in `jsr.js`.
- Welche Aufgaben bei der Client-Server Interaktion übernimmt die `xhr()` Funktion, und welche werden durch die Nutzer dieser Funktion implementiert?
- Wie werden die nutzerspezifischen Aufgaben mit der durch `xhr()` implementierten Funktionalität zusammengeführt?
- Welche Medientypen werden dem Server als verarbeitbar durch den Client angezeigt, und weshalb ist die angegebene Typenmenge erforderlich? Was müsste geändert werden, um hier in Abhängigkeit vom Nutzungsfall eine gezielte Typangabe machen zu können?
- Verfolgen Sie die Logmeldungen bei Ausführung von `xhr()` in Firebug und versuchen Sie, den im Lehrmaterial dargestellten client-seitigen Verarbeitungszyklus von HTTP Requests und Responses anhand der Meldungen für `onreadstatechange` nachzuvollziehen. Weshalb passiert hier nur für den Fall, dass der 'Ready State' den Wert `4` hat, mehr als die Ausgabe einer Logmeldung?
  - den Verarbeitungszyklus finden Sie in `http://moodle.oncampus.de/modules/ir493/onmod/IAMJSL/interakt/xmlhttp.shtml`
- Welche Funktionalität aus der `onreadystatechange` Callback-Funktion könnte bereits für einen früheren 'Ready State' ausgeführt werden, und für welchen?
- Wann und wie wird ein `JSON` Objekt im Request Body übermittelt? (davon machen die vorliegenden Beispiele noch keinen Gebrauch)

AJAX

- Die `create`-Funktionen in `jsr.js` illustrieren die Verwendung der DOM API für den lesenden und schreibenden Zugriff auf das dargestellte HTML Dokument.
- Kommentieren Sie aus der Methode `createVerknuepfungen()` die `while` Schleife aus und laden Sie das HTML Dokument neu. Was ändert sich an der Darstellung und weshalb?
- Betrachten Sie das 'Einführungstext' Element im Firefox Inspector. Weshalb enthält das als `contentfragment` markierte `<div>` Element ein weiteres `<div>` Element?
- Halten Sie es für möglich, dass Performanceunterschiede bei der Verwendung von `querySelector()` / `querySelectorAll()` für den Zugriff auf Elemente bestehen, die alternativ durch `getElementsById()` oder `getElementsByClassName()` ausgelesen werden könnten? Schauen Sie auch *hier* nach...

Sonstiges

Demovideo: `https://connect.oncampus.de/p5i5d9zas6z/`

Ü JSR2 Dynamischer Aufbau von Ansichten (17 Punkte)

Aufgabe

Bauen Sie das in ÜCSS4 entwickelte Gesamtlayout dynamisch auf, analog zu den Implementierungsbeispielen.

Anforderungen

1. Die Konfigurationsdatei, die die Anordnung der Elemente beschreibt, soll in einer `onload` Funktion – wie in den Beispielen – vom Server geladen und dann für den Aufbau der Ansicht verwendet verwendet werden.***(2 P.)***

2. Eine Wechsel der Konfiguration soll für eine ausgewählte Menge von Konfigurationsdateien ohne Eingriff in den Quellcode möglich sein. Verwenden Sie dafür ein Bedienelement, das Ihnen ein einfaches 'Umschalten' zwischen den Elementen einer Menge beliebiger Konfigurationsdateien ermöglicht. Die Ansicht soll jeweils auf Grundlage der aktuellen Konfiguration neu aufgebaut werden. Nichterfüllung dieser Anforderung hat eine pauschale Halbierung der insgesamt in JSR2 erzielten Punktzahl zur Folge.

3. Der Inhalt des 'Textauszug' Elements soll durch eine separate HTML Datei bereit gestellt werden, die in der JSON Beschreibung des Elements referenziert wird. Diese Datei soll keine Überschrift enthalten. Falls Sie JSR2 zusammen mit den vorangegangenen Aufgaben umsetzen, sollen Darstellung und Struktur von 'Textauszug' den Anforderungen aus Übung CSS3 entsprechen.***(3 P.)***

4. Die Konfiguration von Imgbox-Elementen enthält eine Referenz auf das darzustellende Bild sowie ein `description` Attribut mit textuellem Inhalt. Dieser Text soll in einem `alert()` Dialog angezeigt werden, wenn der Nutzer auf den blauen Kringel in der rechten oberen Ecke des Elements klickt. ***(2 P.)***

5. Es soll möglich sein, beliebig viele 'Medienverweise' Element anzuzeigen, die jeweils beliebig viele einzelne Titel enthalten können. ***(5 P.)***

6. Die in blau dargestellte Überschrift des 'Medienverweise' Elements wird aus dem `title` Attribut des JSON Objekts für das betreffenden Element entnommen. ***(1 P.)***

7. Die Elemente sollen in die im Attribut `renderContainer` angegebene Spalte im Spaltenlayout platziert werden (`'left'`, `'middle'` oder `'right'`) ***(2 P.)***

8. Elemente, denen dieselbe Spalte zugewiesen ist, sollen entsprechend der Reihenfolge im JSON Objekt untereinander angeordnet werden. ***(1 P.)***

9. Elemente, die nicht in der Konfiguration angegeben sind, sollen nicht dargestellt werden. ***(1 P.)***

10. Alle anderen Funktionen, die in den Übungen zu CSS und JSL umgesetzt wurden, inklusive des Wechsels in die Detailansicht, soilen weiterhin funktionsfähig sein. Siehe dafür auch die allgemeinen Bearbeitungshinweise.

Bearbeitungshinweise

- Ihnen stehen – im Verzeichnis `data/uebungen` der Implementierungsbeispiele – drei verschiedene JSON Konfigurationsdateien zur Verfügung, die die von Ihnen in den vorangegangenen Aufgaben bearbeiteten Elemente in verschiedener Anordnung enthalten. Bei der Abnahme wird überprüft, ob die Ansichten entsprechend der in der Konfiguration beschriebenen Anordung aufgebaut werden können.
- Die genannten Konfigurationsdateien beschreiben ein 3-spaltiges Layout. Im Ggs. dazu liegt der in den Implementierungsbeispielen verwendeten Datei `die_umsiedlerin.json` ein 6-spaltiges Layout zugrunde, das u.a. die Werte `'centerRight'` und `'centerLeft'` für die Angabe der Spalte im Attribut `renderContainer` verwendet. Für die vorliegende Übungsaufgabe müssen Sie nur das 3-spaltige Layout umsetzen.
- ***Anforderung 2***: Hierfür können Sie die Lösung aus den Implementierungsbeispielen in `jsr.js` übernehmen.
- Das Wechseln der Konfigurationsdatei zur Überprüfung der drei Varianten können Sie manuell umsetzen (d.h. im HTML Dokument ersetzen + Neuladen), oder Sie bauen in Ihre Ansicht einen 'Umschalt-Aktion', bei der die Konfiguration nachgeladen wird und die Ansicht aktualisiert wird. Dafür könnten Sie z.B. die 'Zurück' Aktion in der Kopfzeile verwenden
- Übernehmen Sie diese Konfigurationsdateien sowie das `content` Verzeichnis komplett in Ihr eigenes Projekt.
- Die in den Implementierungsbeispielen enthaltenen JavaScript Dateien können Sie in Ihrem eigenen Projekt wieder verwenden.
- Die in den Konfigurationsdateien referenzierten Bilder, Dokumente etc. können Sie durch Referenzen auf eigene Dateien ersetzen.
- ***Anforderung 3***: Das Laden der Inhalte für `Textauszug` aus einer separaten HTML Datei können Sie analog zur Behandlung von 'Einführungstext' in den Implementierungsbeispielen umsetzen.
- Demovideo: `https://connect.oncampus.de/p3rrfzro5qi/`

NJM

Ü NJM0 Vorbereitende Maßnahmen

- Laden Sie die NodeJS Binaries aus dem Moodle und importieren Sie diese ebenfalls in Eclipse. Es wird ein Projekt `Binrunnable` erstellt.
  - Falls Sie Linux nutzen, dann laden Sie NodeJS von `http://nodejs.org/download/` herunter.
- Laden Sie von `http://www.mongodb.org/downloads` die für Ihre Plattform geeignete MongoDB Version des *Production Releases* herunter und entpacken Sie die Archivdatei in Ihr `Binrunnable` Projekt.
- Überprüfen Sie das für Ihre Plattform zu nutzenden Skript `mdb.sh`/`mdb.bat`. Dieses sollte das `mongod` Executable aus der MongoDB Installation im `Binrunnable` Projekt referenzieren. Nehmen Sie hier ggf. eine Anpassung vor.
- Führen Sie zunächst das `mdb` Skript im Beispielprojekt aus – dafür können Sie die Konsolenansicht von Aptana verwenden. Damit wird der MongoDB Server gestartet.
- Führen Sie dann das `njs` aus. Dieses startet den NodeJS Webserver.
- Öffnen Sie die URL, unter der Ihre Webanwendung in NodeJS gestartet wird, im Browser. Ihnen sollte eine Liste mit Werkstiteln angezeigt werden.
- Wählen Sie einen Titel aus. Ihnen sollte dann eine 'leere Gesamtansicht' für den betreffenden Titel angezeigt werden.
- Falls Sie einen Fehler bekommen, dann ersetzen Sie im Skript `webserver.js` in den Implementierungsbeispielen die `ip` Variable im Aufruf von `server.listen(ip,port` durch den String `127.0.0.1` und starten Sie `njs` neu.

Sonstiges

Demovideo: `https://connect.oncampus.de/p2231hw1w59/`

Ü NJM1 Beispielanwendung

Schauen Sie sich noch einmal das ‘Big Picture’ der Interaktion zwischen den Komponenten der Beispielanwendung in `http://moodle.oncampus.de/modules/ir493/onmod/IAMNJM/nodejs/grund.shtml` an. Versuchen Sie, den kompletten Durchlauf eines Aufrufs einer CRUD Methode nachzuvollziehen, beginnend mit der Ausführung einer Aktion durch den Nutzer auf Ebene der graphischen Nutzeroberfläche. Schauen Sie sich dafür die nachfolgend genannten Skripte und Funktionen an.

`webcontent/js/controller/TopicviewViewController.js` und `TopicviewCRUDOperationsRemote.js`: `createTopicview()`, `readTopicview()`, `updateTopicview()`, `deleteTopicview()`

- Wann wird die Ansicht aktualisiert? Unmittelbar nach der Nutzereingabe oder erst, wenn die Aktion erfolgreich auf Serverseite durchgeführt wurde?

`webserver.js`: Callback-Funktion von `createServer()`

- Auf welcher Grundlage wird entschieden, ob ein HTTP Request mit der Auslieferung eines statischen Dokuments erwidert wird oder ob die Bearbeitung des Requests durch das Skript `http2mdb` erfolgt?
- Was muss auf Seiten von `TopicviewViewController.js` getan werden, damit die CRUD Operationen durch `http2mdb` bearbeitet werden können?

`njsimpl/http2mdb.js`: `processRequest()` sowie `doGet()`, `doPost()`, `doPut()` und `doDelete()`

- nach welchem Kriterium erfolgt die Weiterverarbeitung der Anfrage, die an `processRequest()` übergeben wird?

`http2mdb.js`: `readTopicview()`, `createTopicview()`, etc.

- Wie erfolgt in diesen Funktionen das Zusammenspiel zwischen ggf. dem Auslesen des HTTP Request Body, dem Aufruf der betreffenden CRUD Funktion der MongoDB API, sowie der Erzeugung des HTTP Response?
- In welchen Funktionen liegt eine ‘Verschachtelung’ mehrerer Callback-Funktionen vor, d.h. die Verwendung von Callbacks in Callbacks? Weshalb ist dies an den betreffenden Stellen erforderlich?

Sonstiges

Demovideo: `https://connect.oncampus.de/p3jlxxtcmrg/` (Beispiele zu NJM+FRM)

Ü NJM2 CRUD für 'Imgbox' (10 Punkte)

Aufgabe

Implementieren Sie die Operationen *create*, *update* und *delete* für Imgbox-Elemente und zeigen Sie Imgbox-Elemente in der Topic-Ansicht an. Erforderlich sind hierfür nur client-seitige Implementierungsmaßnahmen. Alle server-seitigen Operationen sind bereits umgesetzt. Diese Aufgabe ist eine 'Vorstufe' für die Aufgabe NJM3, die Sie in NJM3 erweitern werden. Insbesondere werden Sie dort auch die *read* Operation bezüglich der mit Topic-Ansichten assoziierten Imgbox-Elemente umsetzen.

Anforderungen

1. Die Imgbox-Elemente sollen über die folgenden Attribute verfügen: `src`, `title` und `description`. ***(1 P.)***
2. Imgbox-Elemente sollen anhand der server-seitig zugewiesenen `_id` identifiziert werden. ***(1 P.)***
3. Setzen Sie die Operationen durch Ausführung eines `XMLHttpRequest` bezüglich des mittels `webserver.js` gestarteten NodeJS Servers um. ***(3 P.)***
4. Weisen Sie beim Aufruf von *create* einen Wert Ihrer Wahl für `src` zu und weisen Sie bei Ausführung von *update* einen davon verschiedenen Wert zu. ***(1 P.)***
5. Nachdem ein Imgbox-Element server-seitig erfolgreich erstellt worden ist, soll dieses entsprechend dem in Ü CSS3 vorgesehenen Styling auf der Topic-Ansicht dargestellt werden. ***(2 P.)***
6. Nach Ausführung der Operationen *update* und *delete* soll die Darstellung des Imgbox-Elements modifiziert bzw. entfernt werden.***(2 P.)***
7. Die Operationen *update* und *delete* brauchen für diese Aufgabe nur dann ausführbar zu sein, wenn unmittelbar zuvor *create* ausgeführt wurde. Unmittelbar nach einem *reload* der Ansicht oder unmittelbar nach Zugriff aus der Titelliste müssen die Aktionen nicht funktionsfähig sein. Diese Einschränkung werden Sie in NJM3 aufheben.

Bearbeitungshinweise

- Verwenden Sie für die Umsetzung eine Kopie des Beispielprojekts `org.dieschnittstelle.iam.njm_frm_mfm`. In diesem Projekt können Sie dann alle folgenden Aufgaben umsetzen.
- Um die von MongoDB verwendeten Daten Ihrer Anwendung getrennt von den Daten der Implementierungsbeispiele zu handhaben, setzen Sie bitte den Namen Ihres Projekts im `-dbpath` Parameter im `mdb` Skript.
- ***Anforderung 2*** Die `_id` ist Ihnen bis auf weiteres nur dann bekannt, wenn Sie die *create* Operation ausgeführt haben. Bei Zugriff auf die Topic-Ansicht kennen Sie die `_id` noch nicht. Dafür werden Sie in NJM3 Erweiterungen vornehmen.
- ***Anforderung 3***: Zur Auslösung der CRUD Operationen stehen Ihnen bereits Aktionen in der Fußleiste von `topicview.html` zur Verfügung, die Sie bei Betätigung des Pfeils am rechten Rand der Fußleiste angezeigt bekommen.

- Demovideo: https://connect.oncampus.de/p6fgwa4hdzh/ (alle Aufgaben zu NJM+FRM+MFM)

Ü NJM3 Verknüpfung 'Topicview' - 'Imgbox' (9 Punkte)

Aufgabe

Assoziieren Sie Imgbox-Elemente auf Ebene der Datenbank mit den Topic-Ansichten in denen die Imgbox-Elemente erstellt werden. Auch hierfür sind keine server-seitigen Implementierungsmaßnahmen erforderlich.

Anforderungen

1. Erweitern Sie die *create* Operation aus NJM2 wie folgt: falls ein `Imgbox` erfolgreich erstellt wurde, soll der `contentItems` Liste des Topicview, von dem aus `createImgbox()` aufgerufen wurde, eine Referenz der folgenden Form hinzugefügt werden:
   `{type: 'imgbox', renderContainer: 'left', imgboxid: <imgboxid>}`, wobei `<imgboxid>` die `_id` des neu erstellen Imgbox-Elements ist. ***(2 P.)***

2. Setzen Sie die Aktualisierung von `contentItems` so um, dass nur das neu hinzuzufügende Element vom Client an den NodeJS Server und von dort an MongoDB übermittelt wird und nicht das gesamte `Topicview` Objekt aktualisiert wird. ***(2 P.)***

3. Erweitern Sie die *delete* Operation aus NJM2 wie folgt: falls ein Imgbox-Element erfolgreich gelöscht wurde, soll auch die Referenz auf dieses Element aus `contentItems` entfernt werden. Auch hierfür soll nur ein partielles Update durchgeführt werden. ***(2 P.)***

4. Wenn für einen Topicview bereits ein Imgbox-Element existiert, soll dieses bei Darstellung des `topicview.html` Dokuments entsprechend dem dafür vorgesehenen Styling (siehe Ü CSS3) dargestellt werden. ***(2 P.)***

5. Falls ein Imgbox-Element bereits existiert oder neu erstellt wurde, soll bei erneuter Ausführung von `createImgbox()` ein Warnhinweis angezeigt werden und die Imgbox-Erstellung unterbunden werden. ***(1 P.)***

6. Alle anderen Funktionen aus NJM2 sollen weiterhin verfügbar sein. Lediglich die Einschränkung aus ***Anforderung 7*** wird durch die Umsetzung von NJM3 aufgehoben. Falls dies nicht erfüllt ist, wird für NJM3 maximal die Hälfte der möglichen Punktzahl vergeben.

Bearbeitungshinweise

- ***Anforderung 1***, ***Anforderung 2***, ***Anforderung 3*** Die Funktionen in `http2mdb.js` stellen bereits die Funktionalität des partiellen Updates für die Hinzufügung von Elementen zu `contentItems` gemäß ***Anforderung 1*** und das Löschen gemäß ***Anforderung 3*** zur Verfügung. Um auf diese Funktionalität zuzgreifen, müssen Sie lediglich auf Ebene des browser-seitig ausgeführten JavaScript Codes HTTP Requests der folgenden Form initiieren:

  – Hinzufügung: `PUT /topicviews/<topicid>/contentItems` und Übertragung des hinzuzufügenden Elements im Body des Requests. `<topicid>` identifiziert den Topicview, dessen `contentItems` Liste das Element hinzugefügt werden soll.

  – Löschen: `DELETE /topicviews/<topicid>/contentItems/<elementtype>`. Auch hier identifiziert `<topicid>` den Topicview, auf dessen `contentItems` Liste zugegriffen werden soll. `elementtype` bezeichnet den Typ des Elements / der Elemente, die dabei aus `contentItems`

entfernt werden sollen. In Ihrem Fall sieht die URL also wie folgt aus:
`/topicviews/<topicid>/contentItems/imgbox`

- ***Anforderung 3*** Für das Löschen, aber auch für das Auslesen oder die Aktualisierung eines Objekts benötigen Sie `_id` des Objekts, die Sie aus der Objektreferenz aus `contentItems` auslesen können. Dafür steht Ihnen in `TopicviewViewController.js` die Funktion `getObjektIdForTopicview()` zur Verfügung.
- ***Anforderung 4***: Die Funktion zum Auslesen eines evtl. bereits existierenden Imgbox-Elements können Sie aufrufen, nachdem ein Topicview ausgelesen wurde und falls dessen `contentItems` Liste eine Imgbox-Referenz enthält.

FRM

Ü FRM0 Vorbereitende Maßnahmen

- Startansicht der Anwendung ist die Ihnen bereits bekannte Titelliste. Bei Auswahl eines Elements gelangen Sie in die Topic-Ansicht für den ausgewählten Titel.
- Mittels *'Long Press'* auf der Topic-Ansicht oder eines Tap/Click auf die Kopfleiste können Sie das Formular zum Erstellen/Editieren/Löschen des `topicview` Elements für die ausgewählte Ansicht öffnen.

Ü FRM1 Beispielanwendung

Umsetzung des Formulars

- Umsetzung von *Create/Update* (`TitleFormViewController.js`)
  - Wie wird für die Create/Update Aktion die Auslösung der CRUD Datenzugriffe via `XMLHttpRequest` umgesetzt?
  - Welches DOM Ereignis löst den Datenzugriff aus?
  - Wie könnte man das selbe Ereignis auf anderem Wege mit der auszuführenden CRUD Funktion assoziieren?
  - Wo werden die Formulardaten ausgelesen, die für die Ausführung der Funktion verwendet und an den Server übermittelt werden?
- Umsetzung von *Delete* (`TitleFormViewController.js`)
  - Wie wird der *Delete* Datenzugriff durch den an den Server übermittelten HTTP Request identifiziert?
  - Weshalb ist für den Aufruf dieser Funktion kein Formular erforderlich?
- `njsimpl/http2mdb.js`
  - Weshalb bekommen wir einen Fehler, wenn wir aus dem `titleForm` Formular unter der in `<form>` angegebenen `action` via `POST` auf die Webanwendung zugreifen?
  - Weshalb tritt dieser Fehler bei Verwendung von `XMLHttpRequest` nicht auf?

Andere Aspekte der Implementierung

- Öffnen und Schließen der Editieransicht (`uiutils.js`, `EditviewViewController.js`)
  - Wie wird das Öffnen der Editieransicht nach *long press* veranlasst?
  - Weshalb wird die Editieransicht geschlossen, wenn man außerhalb des Tab-Bereichs klickt und anderweitig nicht?
  - Weshalb erfolgt das Schließen im Ggs. zum Einblenden abrupt?
- Bedienung der Tabs (`uiutils.css`, `EditviewViewController.js`, `TitleFormViewController.js`)
  - Beachten Sie, dass die Tab-Ansicht ohne Verwendung von JavaScript und allein mittels HTML und CSS umgesetzt ist. Dies erfolgt auf Basis der Vorschläge in `http://www.sitepoint.com/css3-tabs-using-target-selector/`.
  - Wie wird dafür gesorgt, dass bei Öffnen des Editview sowohl der `Titel`-Tab geöffnet ist, als auch der Focus auf ein Eingabefeld der aktuellen Ansicht gesetzt wird (siehe `openEditview()`)?

Sonstiges

Demovideo: `https://connect.oncampus.de/p3jlxxtcmrg/` (Beispiele zu NJM+FRM)

Ü FRM2 CRUD Formular für Imgbox (21 Punkte)

Aufgabe

Erstellen Sie eine Formularansicht, die Ihnen das *Erstellen*, das *Aktualisieren* und das *Löschen* eines Imgbox-Elements bezüglich der dargestellten Topic-Ansicht ermöglicht.

Anforderungen

1. Die Ansicht soll in der mit `ImgboxTab` markierten `<section>` der Tab-Ansicht dargestellt werden, die in den Implementierungsbeispielen enthalten ist. ***(1 P.)***
2. Die Ansicht soll über ein Bedienelement zum *Erzeugen/Modifizieren* von Imgbox-Elementen verfügen, das als `submit`-Input eines Formulars realisiert wird. ***(1 P.)***
3. Die Ansicht soll über ein weiteres Bedienelement zum *Löschen* von Imgbox-Elementen verfügen. ***(1 P.)***
4. Das Formular zum Erzeugen/Modifizieren soll über geeignete Bedienelemente zur Erstellung und Darstellung der Attribute `src`, `title` und `description` von `Imgbox` verfügen. ***(3 P.)***
5. Falls noch kein `Imgbox`-Element für die dargestellte Topic-Ansicht existiert, dann soll bei Betätigung des Bedienelements aus ***Anforderung 2*** die `createImgbox()` Aktion aus NJM2/3 aufgerufen werden, d.h. das Imgbox-Element soll erstellt und eine Referenz darauf dem `topicview` hinzugefügt werden. ***(3 P.)***
6. Wenn kein `Imgbox`-Element für die Ansicht existiert, dann soll das Bedienelements aus ***Anforderung 3*** nicht bedienbar sein. ***(2 P.)***
7. Wenn ein `Imgbox`-Element für die Ansicht existiert, dann sollen bei Öffnen der Tab-Ansicht die Tabs für `imgboxTab` und `imgboxlistTab` bereits vorhanden sein. Existiert kein Imgbox-Element, sind die Tabs nicht sichtbar. ***(2 P.)***
8. Existiert bereits ein `Imgbox`-Element für die dargestellte Topic-Ansicht, dann sollen dessen Attribute in den Bedienelementen des Formulars aus ***Anforderung 4*** dargestellt werden. ***(3 P.)***
9. Existiert für eine Topic-Ansicht noch kein `Imgbox-Element`, dann soll bei erstmaligem Öffnen der Editieransicht der Tab für das Imgbox-Formular nicht dargestellt werden. ***(1 P.)***
10. Wenn ein `Imgbox`-Element für die Ansicht existiert, dann soll bei Betätigung des Bedienelements aus ***Anforderung 2*** die `updateImgbox()` Aktion aus NJM2/3 ausgeführt werden. ***(2 P.)***
11. Wenn ein `Imgbox`-Element für die Ansicht existiert, dann soll bei Betätigung des Bedienelements aus ***Anforderung 3*** die `deleteImgbox()` Aktion aus NJM2/3 ausgeführt werden. ***(2 P.)***

Bearbeitungshinweise

- Für die Umsetzung der Controller-Funktionalität des Imgbox-Formulars können Sie das Codegerüst `ImgboxFormViewController.js` verwenden.

- Sie können sich bei der Umsetzung an der Handhabung der CRUD Operationen für `topicvdiew` in den Implementierungsbeispielen orientieren.
- Alle HTML-Bestandteile, CSS Regeln und JavaScript Funktionen können Sie aus den Implementierungsbeispielen übernehmen.
- Zur Erleicherung der Implementierung können Sie das `editview <div>` Element zunächst in Ihrem HTML Dokument mit `class="overlay"` markieren und die Tabs für `Imgbox` darin einfügen.
- ***Anforderung 6***: Dafür können Sie das `disabled` Attribut auf dem Bedienelement auf `disabled=true` setzen, wie dies auch in den Implementierungsbeispielen gemacht wird.
- Demovideo: `https://connect.oncampus.de/p6fgwa4hdzh/` (alle Aufgaben zu NJM+FRM+MFM)

Ü FRM3 Imgbox-Liste (12 Punkte)

Aufgabe

Erstellen Sie eine Ansicht, die alle existierenden Imgbox-Elemente der `imgboxs` Collection in einer Tabellenansicht darstellt.

Anforderungen

1. Die Tabelle soll in der mit `imgboxlistTab` markierten `<section>` der Tab-Ansicht dargestellt werden, die in den Implementierungsbeispielen verwendet wird. ***(1 P.)***
2. Die Tabelle soll die Menge aller `Imgbox` Elemente der `imgboxs` Collection anzeigen, wobei für jedes Imgbox-Element eine Zeile verwendet wird und die Werte der Attribute `src`, `title`, `description` in jeweils einer Spalte dargestellt werden. ***(3 P.)***
3. Zur Darstellung des `src` Attributs soll ein `<img>` Element verwendet werden, in dem das durch das `src` Attribut referenzierte Bild in angemessener Verkleinerung dargestellt wird. ***(2 P.)***
4. Die Breite der Tab-Ansicht soll durch die Tabelle nicht überschritten werden. Falls die Höhe der Tabelle die Höhe der Tab-Ansicht überschreitet, soll vertikales Scrolling möglich sein. ***(2 P.)***
5. Wird über das Formular für Imgbox-Elemente ein Element gelöscht, geändert oder neu hinzugefügt, soll die Tab Ansicht geöffnet bleiben und die Tabelle mit der Imgbox-Liste entsprechend modifiziert werden, ohne dass die gesamte Imgbox-Liste neu geladen wird. ***(4 P.)***

Bearbeitungshinweise

- Für die Umsetzung der Controller-Funktionalität der Imgbox-Liste können Sie das Codegerüst `ImgboxlistViewController.js` verwenden.
- Hinweise zur Erstellung von Tabellen in HTML finden Sie im Skript und in den von dort referenzierten Quellen.
- ***Anforderung 2***: Spaltenüberschriften sind nicht erforderlich.
- Falls Sie ein stärker an Gestaltungsmerkmalen von Smartphones orientiertes Design verwenden, können Sie anstelle einer Tabelle auch eine Listenansicht umsetzen, bei der `title` und `description` in einer Spalte untereinander angeordnet werden.

MFM

Ü MFM1 Beispielanwendung

`EinfuehrungstextFormViewController.js/initialiseView()`

- In der Variable `radioOnClickListener` wird eine Funktion definiert, die auf die Betätigung eines Radiobuttons reagieren soll. Wie wird die Funktion den drei Radiobuttons zugewiesen und wie wird ermittelt, für welchen Button sie aufgerufen wird?
- In der Variable `inputOnInvalidListener` wird eine Funktion definiert, die auf das `invalid` Ereignis bezüglich mehrerer Eingabeelemente des Formulars reagieren soll. Welchen Eingabeelementen wird die Funktion zugewiesen?
- Ein direktes Auslesen des Werts eines Formularfelds, dessen mögliche Werte durch eine Menge von Radiobutton Elementen auswählbar sind, aus der Formularrepräsentation in JavaScript scheint nach Konsultation diverser Entwicklerforen nicht möglich zu sein. Wie wird im `onsubmit` Handler für das Formular `einfuehrungstextForm` ermittelt, ob ein Radiobutton ausgewählt wurde?
- Weshalb ist es in der Implementierung von `radioOnClickListener` nicht erforderlich, das `required` Attribut auf `inputSrc` bzw. `inputTxt` bei Deaktivierung des betreffenden Elements zu entfernen?

`EinfuehrungstextFormViewController.js/submitEinfuehrungstextForm()`

- Weshalb wird im Fall der Option `fileupload` im Ggs. zu den anderen Fällen der Wert `true` zurück gegeben und was hat dies zur Folge?
- Wie wird in der Behandlung der Option `fileuploadFormdata` auf die ausgewählte Datei zugegriffen, und weshalb kann hier davon ausgegangen werden, dass eine Datei ausgewählt wurde?

`topicview.html` und `EinfuehrungstextFormViewController.js`

- Wie wird das `<output>` Element `validationMessages` befüllt?
- Für welchen Fall bestimmen wir als Entwickler, welcher Text angezeigt werden soll?
- Wie wird im Fall einer Texteingabe oder im Fall der Änderung einer Auswahloption dafür gesorgt, dass eine ggf. angezeigte Fehlermeldung verschwindet?

`editview.css`

- Wie wird verhindert, dass die rote Default-Umrandung für als ungültig validierte Felder angezeigt wird?
- Wie wird das ‘Ausgrauen’ deaktivierter Eingabeelemente bewirkt?

`TopicviewViewController.js/onMultipartResponse()`

- Wer ruft diese Funktion auf und wann? Suchen Sie, falls erforderlich, im gesamten Projekt nach dem String `onMultipartResponse` (ohne Klammern).
- Wie wird der Funktion das JSON Objekt als Argument übergeben?

Sonstiges

Demovideo: https://connect.oncampus.de/p6ai8rpsdep/

Ü MFM2 Multipart Formular für Imgbox (22 Punkte)

Aufgabe

Erweitern Sie das CRUD Formular für Imgbox aus FRM2 um die Optionen, das anzuzeigende Bild durch Dateiauswahl hochzuladen oder es aus der Imgbox-Liste auszuwählen.

Anforderungen

1. Zur Auswahl der Optionen 'URL', 'Upload' und 'Liste' sollen Radio Buttons verwendet werden. In der nächsten Lerneinheit werden wir eine weitere Option hinzufügen. ***(2 P.)***
2. Wird die Option 'URL' ausgewählt, dann soll die Eingabe einer URL obligatorisch sein. ***(2 P.)***
3. Nur wohlgeformte URLs sollen verarbeitet werden. ***(1 P.)***
4. Wird die Option 'Upload' ausgewählt, dann soll die Auswahl einer Bilddatei zum Upload obligatorisch sein. ***(1 P.)***
5. Wird die Option 'Liste' ausgewählt, dann soll die Auswahl eines Imgbox-Elements aus der Imgbox-Liste obligatorisch sein. ***(1 P.)***
6. Bei Auswahl der Option 'Liste' soll automatisch der Tab 'Imgbox-Liste' in den Vordergrund rücken. Alle weiteren Anforderungen zur Imgbox-Auswahl sind in Ü MFM3 formuliert. ***(2 P.)***
7. Die Angabe eines Titels soll obligatorisch sein. ***(1 P.)***
8. Die Angabe einer Description soll optional sein. ***(1 P.)***
9. Ein Absenden des Formulars ohne Auswahl einer Option soll nicht möglich sein. ***(1 P.)***
10. Falls die Option 'URL' ausgewählt ist, soll bei Absenden des Formulars die Erstellung bzw. die Aktualisierung des Imgbox-Elements wie in Ü FRM2 erfolgen. Ein Wechsel zwischen den verschiedenen Erstellungsoptionen 'URL', 'Upload' und 'Liste' braucht für die Aktuaisierung von Imgbox-Elementen nicht berücksichtigt zu werden. ***(1 P.)***
11. Falls die Option 'Liste' ausgewählt ist, soll bei Absenden des Formulars nur eine entsprechende Imgbox-Referenz zum dargestellten `Topicview` hinzugefügt werden. ***(2 P.)***
12. Falls die Option 'Upload' ausgewählt ist, soll bei Absenden des Formulars die ausgewählte Bilddatei auf den Server hochgeladen werden. ***(4 P.)***
13. Mit dem Rückgabewert des Uploads aus ***Anforderung 12*** soll wie in FRM2 ein neues Imgbox-Element erstellt werden. ***(3 P.)***

Bearbeitungshinweise

- ***Anforderung 3***: Sie können dafür die in HTML vorgesehene Gültigkeitsprüfung nutzen
- ***Anforderung 9***: Sie können dafür eine geeignete Option vor-auswählen.
- ***Anforderung 6***: Dafür können Sie die Funktion `showTabForElementtype()` in `EditviewViewController.js` verwenden.

- ***Anforderung 11***: Das Imgbox-Element braucht nicht erstellt zu werden, da es bei Auswahl aus der Liste ja bereits existiert.
- ***Anforderung 12***: Sie können dafür genauso vorgehen, wie es die Implementierungsbeispiele für den Fall von ‘Einführungstext’ zeigen. Server-seitige Programmierung ist nicht erforderlich.
- Demovideo: `https://connect.oncampus.de/p6fgwa4hdzh/` (alle Aufgaben zu NJM+FRM+MFM)

Ü MFM3 Auswahl von Imgbox-Elementen aus der Liste (13 Punkte)

Aufgabe

Erweitern Sie die Imgbox-Liste aus FRM3 um eine Auswahlmöglichkeit, mit der die Option ‘Liste’ aus MFM2 umgesetzt wird.

Anforderungen

1. Erweitern Sie die Tabelle zur Darstellung der Imgbox-Elemente um eine Spalte, die für jedes Imgbox-Element ein Auswahlelement zur Auswahl des betreffenden Elements enthält. ***(3 P.)***
2. Eine Auswahl soll nur möglich sein, wenn im Formular aus MFM2 die Option ‘Liste’ ausgewählt ist. ***(2 P.)***
3. Es soll immer nur ein Imgbox-Element ausgewählt werden können. ***(2 P.)***
4. Wird ein Imgbox-Element ausgewählt, dann soll der Tab mit dem CRUD Formular für Imgbox-Elemente in den Vordergrund gebracht werden. ***(3 P.)***
5. Die Attribute des Imgbox-Elements sollen dann in den entsprechenden Feldern des Formulars *nicht editierbar* (‘ausgegraut’) angezeigt werden. ***(3 P.)***

Bearbeitungshinweise

- ***Anforderung 2***: d.h. auch wenn der Nutzer aktiv den Tab mit der Imgbox-Liste auswählt, soll eine Imgbox-Auswahl nur möglich sein, wenn die Option ‘Liste’ ausgewählt ist.
- ***Anforderung 3***: ziehen Sie dafür die Verwendung einer Menge von Radiobuttons mit gleichem `name` Attribut sowie das Einbetten der Tabelle in ein `form` Element in Betracht.
- ***Anforderung 5***: zu diesem Zweck müssen Sie das ausgewählte Imgbox-Element dem Formular in geeigneter Form zur Verfügung stellen.
- Den Radiobuttons können Sie wie in den Implementierungsbeispielen einen gemeinsamen Event Handler zuweisen, in dem Sie überprüfen, für welches Imgbox-Element die Auswahl erfolgt. Weisen Sie den Radiobuttons dafür z.B. eine geeignete `id` zu.

MME

Ü MME1 Beispielanwendung

audioplayer.html/videoplayer.html

- Fügen Sie die Booleschen Attribute `autoplay` und `controls` zu den Multimedia-Elementen hinzu und überprüfen Sie deren Funktion durch Neuladen der jeweiligen HTML Dokumente.

mediacontrols.js.togglePlay()

- Was bewirkt die Setzung von `media.currentTime = 0;`?

mediacontrols.js.mediaLoaded()

- Wo wird festgelegt, dass die Funktion `mediaLoaded()` aufgerufen werden soll?
- Warum wird in `mediaLoaded()` evtl. `initialiseView()` aufgerufen?
- Was passiert, wenn dieser Aufruf auskommentiert wird?

mediainput.js

- Welche Aufgabe hat die erste Codezeile, in der `navigator.getMedia` instantiiert wird?
- Was ist der Wert von `navigator.getMedia` für den Fall, dass der Browser Media Capture unterstützt?

mediainput.js.storeMedia()

- Wie wird der Inhalt des `<video>` Elements auf das `<canvas>` Element übertragen?
- Wie könnte hier eine vergrößerte Darstellung des Videobilds im `<canvas>` mit vierfacher Bildfläche herbeigeführt werden?

mediainput.js.captureAndPlaybackMedia()

- Weshalb wird bei Auswahl von *Video* immer eine Bildaufzeichnung mit Ton angeboten?
- Was müssen Sie tun, um nur eine Bildaufzeichnung zu initiieren?
- Welche Anweisung führt dazu, dass der Kamerastream im `<video>`-Element dargestellt wird?

Sonstiges

Demovideo: `https://connect.oncampus.de/p2hbrf9uigf/`

Ü MME2a Aufnahme eines Bildes für Imgbox (24 Punkte)

Aufgabe

Erweitern Sie das CRUD Formular für Imgbox-Elemente um eine Option, die dem Nutzer die Aufnahme eines Bildes ermöglicht, und erstellen Sie das Imgbox-Element unter Verwendung des aufgenommenen Bildes.

Anforderungen

1. Zur Bildaufnahme soll die Media Capture API verwendet werden. ***(2 P.)***
2. Die Bildaufnahme soll als 'Kamera-Ansicht' umgesetzt werden, die die Editier-Ansicht mit Tab, Formular etc. überlagert. ***(2 P.)***
3. Es soll dafür kein Neuladen eines HTML Dokuments erfolgen. ***(2 P.)***
4. Nach Aufnahme des Bildes soll dieses dem Nutzer angezeigt werden. ***(3 P.)***
5. Der Nutzer soll dann auswählen können, ob er das Bild übernehmen oder verwerfen möchte. ***(2 P.)***
6. Übernimmt der Nutzer das Bild, dann wird die Kamera-Ansicht ausgeblendet und das Formular wieder sichtbar. Die Bilddaten werden dem Formular in geeigneter Form übergeben. ***(3 P.)***
7. Übernimmt der Nutzer das Bild nicht, dann wird die Kamera-Ansicht ausgeblendet und das Formular wird ohne weitere Datenübergabe sichtbar. ***(1 P.)***
8. Im Ggs. zu den Implementierungsbeispielen soll das `<canvas>` Element, in dem das aufgenommene Bild angezeigt wird, deckungsgleich mit dem `<video>` Element sein und dieses verdecken. ***(3 P.)***
9. Bei Erstellung oder Aktualisierung des Imgbox-Elements werden die neu aufgenommenen Bilddaten übertragen und server-seitig in der MongoDB persistiert. ***(3 P.)***
10. Alle implementierten Funktionen bezüglich Imgbox sind in vollem Umfang auch für aufgenommene Bilder funktionsfähig. ***(3 P.)***

Bearbeitungshinweise

- ***Anforderung 8***: dafür können `<canvas>` und `<video>` mit `position:absolute;` und identischem Ursprung positioniert werden. Der sichtbare 'Vordergrund' ist u.a. mittels Verwendung der Style-Property `z-index` kontrollierbar.
- ***Anforderung 9***: Sie können das Imgbox-Element mit einer Data URL als `src` übertragen und persistieren. Dafür ist kein Multipart-Request erforderlich.
- Sie brauchen nur den Fall der Imgbox-*Erstellung* mit Kamerabild zu berücksichtigen. Es ist nicht erforderlich, dass einem bereits mit einem 'normalen' Bild erstellten Imgbox-Element im Rahmen eines Updates ein Kamerabild zugewiesen werden kann.

Ü MME2b Verwendung eines Videos für Imgbox (22 Punkte)

Aufgabe

Erlauben Sie das Hochladen von Videos anstelle von Bildern für Imgbox-Elemente und erstellen Sie das Imgbox-Element mit dem Startbild des Videos.

Anforderungen

1. Zur Umsetzung der Funktionalität soll auf Ebene des Formulars kein neues Bedienelement eingeführt werden. ***(2 P.)***
2. Die 'Extraktion' des Startbilds aus dem Video soll client-seitig nach erfolgreichem Hochladen des Videos erfolgen. ***(10 P.)***
3. Nach der Extraktion soll das Imgbox-Element mit dem extrahierten Startbild erstellt werden. ***(3 P.)***
4. Der Extraktionsvorgang darf keine Eingabe des Nutzers erfordern, darf für den Nutzer aber sichtbar sein. ***(1 P.)***
5. Im Ggs. zu den Implementierungsbeispielen soll das für die Extraktion verwendete `<canvas>` Element, in dem das aufgenommene Bild angezeigt wird, deckungsgleich mit dem zu verwendenden `<video>` Element sein und dieses verdecken. ***(3 P.)***
6. Alle implementierten Funktionen bezüglich Imgbox sind in vollem Umfang auch für Imgbox-Elemente mit Video funktionsfähig. ***(3 P.)***

Bearbeitungshinweise

- Nutzen Sie für die 'Extraktion' des Startbilds aus dem Video dasselbe Verfahren, das wir für die Aufnahme eines Standbilds mit der Kamera verwenden.
- ***Anforderung 3***: wenn Sie ein Video hochladen, dann wird Ihnen im Rückgabeobjekt des Servers der Medientyp als Wert des `mediatype` Attributs angegeben. Davon können Sie abhängig machen, ob Sie eine Extraktion vornehmen oder nicht.
- ***Anforderung 4*** und ***Anforderung 3***: Zur Umsetzung dieser Anforderungen können Sie für die Erstellung einen `XMLHttpRequest` mit `FormData` verwenden.
- ***Anforderung 5***: dafür können `<canvas>` und `<video>` mit `position:absolute;` und identischem Ursprung positioniert werden. Der sichtbare 'Vordergrund' ist u.a. mittels Verwendung der Style-Property `z-index` kontrollierbar.
- Sie brauchen nur den Fall der Imgbox-*Erstellung* mit Video zu berücksichtigen. Es ist nicht erforderlich, dass einem bereits mit einem 'normalen' Bild erstellten Imgbox-Element im Rahmen eines Updates ein Video zugewiesen werden kann.

Ü MME3 Abspielen von Videos (11 Punkte)

Aufgabe

Bei Klick/Touch auf die Bilddarstellung von Imgbox-Elementen soll die bestehende Topicview-Ansicht durch eine Videoplayer-Ansicht ersetzt werden.

Anforderungen

1. Das Abspielen des Videos soll keine weitere Nutzereingabe erfordern. ***(2 P.)***
2. Der Videoplayer soll die Standard-Kontrollleiste des Browsers verwenden. ***(1 P.)***
3. Die Hintergrundfarbe des Videoplayers soll Schwarz sein. ***(1 P.)***
4. Die Videoplayer-Ansicht soll nur den Hauptbereich der Topicview-Ansicht ersetzen. Header und Footer sollen unverändert bleiben. ***(2 P.)***
5. Bei Betätigung des 'Zurück'-Buttons soll das Abspielen des Videos gestoppt und wieder die Topicview-Ansicht angezeigt werden. ***(2 P.)***
6. Der Übergang zum Videoplayer soll mit Aus- und Einblenden umgesetzt werden. ***(2 P.)***
7. Der Übergang zwischen den Ansichten soll ohne Neuladen von HTML Dokumenten umgesetzt werden. ***(1 P.)***

Bearbeitungshinweise

- Sie können ein geeignetes Video Ihrer Wahl verwenden.

LDS

Ü LDS1 Beispielanwendung

Die Error-Meldungen, die Ihnen Aptana evtl. für die Datei `js/lib/ixdb.js` anzeigt, können Sie auf Defizite des JavaScript Editors zurückführen.

`ListviewViewController.js/showItemDialog(), mwf.css`

- Wie wird bei Einblenden eines Popup-Dialogs der Übergang der Listenansicht in den Hintergrund bewirkt?
- Welche Ausdrucksmittel von CSS und JavaScript werden hierfür eingesetzt?

`lListviewViewController.js/addNewItemViewToListView()`

- Wie werden die einzelnen Listenelemente aufgebaut und mit den darzustellenden Daten befüllt?
- Ändern Sie die Wertzuweisung für `newItemView` wie folgt:
  ```
  var newItemView = itemView;
  ```
- Laden Sie die Ansicht neu. Wie hat sich die Darstellung geändert und weshalb?
- Was können Sie daraus hinsichtlich der Funktionalität von `cloneNode()` folgern?

`ListviewViewController.js/deleteItem()/updateItem()`

- Wie wird das aus der Listenansicht zu entfernende bzw. zu aktualisierende Element identifiziert?
- Wo wird die Voraussetzung dafür geschaffen, dass diese Identifikation möglich ist?

Local Storage

- Öffnen Sie die Beispielanwendung, geben Sie `localStorage` auf der Console in Firebug ein und führen Sie *Run* aus.
- Welche Inhalte werden Ihnen für Ihr `localStorage` angezeigt?
- Falls Inhalte aus anderen Beispielanwendungen enthalten sind, weshalb ist dies der Fall?

`lib/ixdb.js/readAllObjects()`

- Wie wird der Array der ausgelesenen Objekte aufgebaut und wann erfolgt deren Darstellung in der Listenansicht?
- Welche Änderung müsste vorgenommen werden, um die Objekte einzeln unmittelbar nach Auslesen eines Objekts darzustellen?

`DataItemCRUDOperations.js/readAllItems()`

- Weshalb wird in der abschließenden `for`-Schleife die Variable `countdown` verwendet, um den Abschluss der Iteration zu detektieren?
- Rufen Sie die Logmeldung 'all objects have been read out!' alternativ nach Abschluss der `for`-Schleife auf und schauen Sie im Log, wann die Ausgabe der Meldung erfolgt.
- Weshalb wird nun zuerst diese Logmeldung ausgegeben und erst danach die Meldungen bezüglich der ausgelesenen Objekte?

Sonstiges

Demovideo: `https://connect.oncampus.de/p65wcnhg2kb/`

Ü LDS2 Verwendung von IndexedDB für Topicview und Imgbox (20 Punkte)

Aufgabe

Erweitern Sie die in den vorangegangenen Aufgaben entwickelte Implementierung durch einen lokalen Datenspeicher mit IndexedDB und ermöglichen Sie eine (ansatzweise) Synchronisierung von server-seitiger und lokaler Datenhaltung

Anforderungen

1. Erstellen Sie eine alternative Implementierung aller bisher verwendeten CRUD Operationen für Topicview und Imgbox, bei der anstelle des Zugriff auf eine server-seitige MongoDB Datenbank eine lokale IndexedDB Datenbank verwendet wird. Die Punkte werden wie folgt vergeben:
   - CRUD für Topicview ***(4 P.)***
   - CRUD für Imgbox ohne Verknüpfung mit Topicview ***(4 P.)***
   - Erstellen und Löschen der Verknüpfung von Topicview und Imgbox ***(3 P.)***
   - Lesen aller Imgbox-Elemente ***(1 P.)***
2. Als Identifikatoren sollen durch Ihre Anwendung zugewiesene IDs und nicht die IDs von IndexedDB verwendet werden. ***(1 P.)***
3. Erstellen Sie eine weitere Implementierung der CRUD Operationen für Topicview und Imgbox, die die lokale Implementierung aus ***Anforderung 1*** sowie die in den vorangegangenen Aufgaben genutzte Implementierung mit server-seitigem Zugriff verwendet. Wird eine schreibende CRUD Operation server-seitig erfolgreich ausgeführt, soll die betreffende Operation auch lokal ausgeführt werden. Als IDs für IndexedDB sollen die IDs aus dem server-seitigen Zugriff genutzt werden. Die Punkte werden wie folgt vergeben:
   - C(R)UD für Topicview ***(3 P.)***
   - C(R)UD für Imgbox mit Erstellen und Löschen von Verknüpfungen ***(3 P.)***
4. Ermöglichen Sie in Ihrer Anwendung das 'Umschalten' zwischen den gemäß ***Anforderung 1*** und ***Anforderung 3*** entwickelten lokalen bzw. 'synchronisierten' Datenzugriffsoperationen und der bisher verwendeten Implementierung mit ausschließlich remote Datenzugriff. ***(1 P.)***
5. Falls ***Anforderung 4*** nicht umgesetzt wird, wird für ***Anforderung 1***-***Anforderung 3*** maximal die Hälfte der genannten Punktzahl vergeben.

Bearbeitungshinweise

- ***Anforderung 1***-***Anforderung 3***: Für die Umsetzung der lokalen und synchronisierten Datenzugriffsoperationen können Sie die Codegerüste `TopicviewCRUDOperationsLocal.js` und `TopicviewCRUDOperationsSynced.js` verwenden. In `TopicviewCRUDOperationsLocal.js` ist bereits die Verwendung der Bibliothek `lib/indexeddb.js` vorgesehen, die generische Funktionen für den Zugriff auf IndexedDB bereit stellt.

- ***Anforderung 3***: Ein Beispiel für die generische Umsetzung dieser Anforderung können Sie der `RemoteMasterSyncedCRUDOperationsImpl.js` Implementierung aus OFF entnehmen.
- ***Anforderung 4***: Diese Funktionalität ist in den aktuellen Implementierungsbeispielen bereits umgesetzt.

OFF

Ü OFF1 Beispielanwendung

Die Error-Meldungen, die Ihnen Aptana evtl. für die Datei `js/lib/IndexedDBCRUDOperationsImpl.js` anzeigt, können Sie darauf zurückführen, dass die IndexedDB API der Entwicklungsumgebung unbekannt ist.

`off.manifest`

- Welche Funktionalität wird durch die Aufrufe der URLs mit dem Segment `http2mdb` erbracht und wo ist diese Funktionalität in der vorliegenden Applikation implementiert?
- Wo wird auf die URL `available` zugegriffen?
- Weshalb ist die Deklaration dieser beiden URLs als `NETWORK` URLs erforderlich?

`js/lib/webapp.js`

- Wie kann die Implementierung auf das Ergebnis der beiden Funktionsaufrufe bezüglich `checkInstalled()` und `install()` auf `navigator.mozApps` reagieren?
- Welche beiden JavaScript APIs, die in der Anwendung verwendet werden, erfordern ein dem entsprechendes Implementierungsmuster, um auf das Resultat eines Aufrufs zuzugreifen?

`js/DataItemCRUDOperations.js`

- Welche Aufgabe kommt der Variablen `crudimpl` zu und wie wird die Variable instantiiert?

`js/lib/*CRUDOperationsImpl.js`

- Wo werden die Werte für die Argumente `onsuccess` und `onerror`, die den meisten Funktionen übergeben werden, gesetzt?
- Weshalb ist die hier umgesetzte Form der Entgegennahme von Funktionsergebnissen erforderlich?

`js/lib/RemoteMasterSyncedCRUDOperationsImpl.js`

- Wozu dient die `countdown` Variable, die an verschiedenen Stellen in der Funktion `syncObjects` verwendet wird? Weshalb ist sie hier erforderlich? (siehe dazu auch Ü LDS1)
- Welchen Funktionen wird in `syncObjects()` das Argument `{object: object}` übergeben, und wie wird dieses Argument von den betreffenden Funktionen verwendet?
- An welcher Stelle im Programmcode wird das besagte als Argument übergebene Objekt `{object: object}` tatsächlich verwendet?
- Weshalb ist diese Form der Übergabe von Werten ‘innerhalb einer Funktion’ im vorliegenden Fall erforderlich? Wäre es auch möglich den Wert der Variable `object` direkt auszuwerten, ohne ihn auf die vorliegende Art ‘durch einen Funktionsaufruf hindurch’ zu übergeben?

Sonstiges

Demovideo: https://connect.oncampus.de/p2u351gcjux/

Ü OFF2 Offline-Anwendung (8 Punkte)

Aufgabe

Machen Sie die in NJM3-LDS2 entwickelte Anwendung als Offline-Webanwendung mit lokaler Datenspeicherung verfügbar. ***(5 P.)***

Anforderungen

1. Unter 'Offline-Betrieb' wird der Zugriff auf die Anwendung ohne laufenden NodeJS Webserver, aber mit verfügbarer Internet-Verbindung verstanden.
2. Für die Ausführung von CRUD Operationen soll die in LDS2 entwickelte Implementierung mit IndexedDB verwendet werden.
3. Falls beim Starten der Anwendung kein Zugriff auf den NodeJS Server möglich ist, soll die lokale CRUD Implementierung automatisch ausgewählt werden. ***(3 P.)***
4. In der Anwendung selbst auf den Webserver hochgeladene Bilder, die von Imgbox-Elementen verwendet werden, brauchen nicht offline verfügbar zu sein.
5. Der Upload von Bildern braucht im Offline-Betrieb ebenfalls nicht funktionsfähig zu sein.

Bearbeitungshinweise

- Diese Aufgabe basiert im wesentlichen auf der Umsetzung von LDS2. Hier geht es dann vor allem noch darum, dass Sie das Offlline-Manifest für Ihre Anwendung erstellen und verwenden.
- ***Anforderung 4***, ***Anforderung 5***: Getestet wird der Offline-Betrieb für Imgbox-Elemente, die externe Bilder via deren URL referenzieren.