Das DOM und jQuery

Inhalt:

HTML-Seiten mithilfe von JavaScript durch die Bearbeitung von DOM-Elementen aktualisieren. Mit jQuery diese sogar animieren.

1)DOM erklären

- Ein Element mit getElementById auswählen
- Übung: Überschrift mithilfe des DOM ersetzten

2) jQuery und die Arbeit mit dem DOM-Baum

- jQuery in die HTML-Website laden

2a)Den Überschrifttext mit jQuery ersetzten – text()

2b)Neue Elemente mit jQuery erzeugen – append()

2c)append mit einer FOR-Schleife nutzen

3)Elemente mit jQuery animieren – fadeOut()

3a)jQuery-Animationen verketten

fadeIn(), slideUp(), slideDown()

Das DOM und jQuery

Das DOM stellt eine Abstraktion des Dokuments dar, die der Browser beim Einlesen des HTML-Quelltextes im Arbeitsspeicher erzeugt. Eigentlich existiert es also rein virtuell.

Mit dem DOM oder Document Object Model kann JavaScript auf dem Inhalt einer Webseite zugreifen. jQuery wiederum ist ein sehr nützliches Werkzeug, welches die Arbeit mit dem DOM erleichtert. Es liefert eine Reihe Funktionen, mit denen man auswählen kann, welche Elemente man nutzen möchte um sie anschließend zu ändern. So kann man z.B. DOM-Elemente wie <h1>, <body> animieren oder ein- und ausblenden.

1)Die HTML-Elemente im DOM-Baum

- Der sogenannte DOM-Baum bildet die Basis einer jeden Webseite.
- Der DOM ist sozusagen ein Abbild der aktuellen HTML-Seite.

Eine HTML-Datei ist im Grunde genommen nichts anderes als eine hierarchische Verschachtelung von HTML-Elementen. Wenn ein Browser vom Webserver einen Quelltext bekommt, versucht er zunächst, sich eine Übersicht über diese Hierarchie zu verschaffen, und erstellt dazu ein Modell, das ***Document Object Model*** **(abgekürzt DOM**) genannt wird, weil es ein Modell der Objekte, also der Dinge auf einer Webseite ist.

Der Dom ist eine Abstraktion folgender Informationen:

- Wie ist der hierarchische Zusammenhang in der Struktur?
- Welches Element ist an welcher Position der Hierarchie?
- Welche Eigenschaften hat es (z.B. Attribute, Inhalte, Nachbarelemente)?

Daraus ergibt sich ein Vorteil gegenüber seriellem Quelltext, nämlich dass bei einer Entnahme oder dem Hinzufügen durch z.B. jQuery oder JavaScript immer mit vollständigen Strukturen gearbeitet werden kann. Ein Dokument behält somit stets seine „Wohlgeformtheit“, also die Art von regelmäßiger Struktur, die in der XML-Datenverarbeitung gefordert ist.

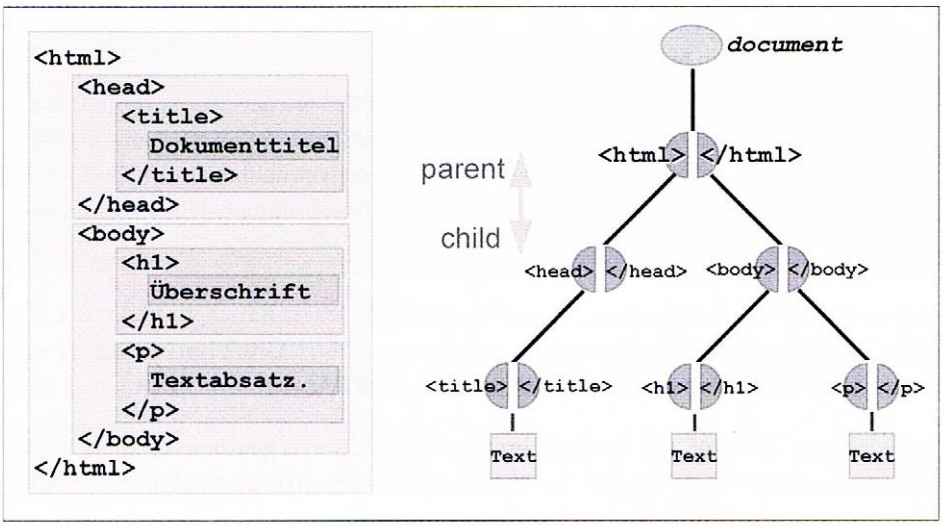
Nach der Darstellung des DOMs (der „nackten Struktur“) liest der Browser alle CSS-Informationen ein („decorating the tree“) und löst dabei auftretende Konflikte und Unstimmigkeiten auf. Anschließend liegt das so genannte „Stylesheet“ vor, nämlich die vollständige Präsentationsvorschrift, die auf das Dokument angewendet werden soll.

Facit:
Der dekorierte Baum stellt ein Abbild des Dokuments im Arbeitsspeicher des Browsers dar, das der Darstellung des Dokuments im Viewport entspricht. Da es sich um eine rein virtuelle Sache handelt, ist dieses Abbild per Programmierung beliebig manipulierbar. Genau dies ist überhaupt die Aufgabe des DOM – eine Schnittstelle (API) zu bieten, die es ermöglicht, mittels einer Programmiersprache auf das Dokument einzuwirken.

Der DOM sieht eigentlich aus wie ein **umgekehrter Baum**, der nach unten hängt, der an einem Punkt aufgehängt ist und sich ab dort verzweigt.

Inzwischen ist der **DOM-Inspector** ein fester Bestandteil vieler Browser.
Um sich ein Bild von diesem Modell zu machen, klicke im Firefox oder im Chrome mit der rechten Maustaste irgendwo ins Browserfenster und wähle im Kontextmenü den Befehl Element untersuchen. Daraufhin erscheinen in der unteren Hälfte des Browserfensters die Entwickler-Tools.

Beispiel 1:
Ganz oben ist der erste Dokumentknoten (DocumentNode), der hier als „document“ bezeichnet wird. Dieser wird vom Browser beim Einlesen (Parsing) erzeugt. Er hängt dann für jedes Element, das er in Quelltextreihenfolge antrifft, einen weiteren Knoten unten an diesen Dokumentenknoten an. Hier spaltet sich der Baum in zwei Zweige, die den <head> und den <body> repräsentieren. Enthält ein Element einen Inhalt, fächert sich der Baum weiter nach unten auf und es entsteht für jeden Bestandteil dieses Inhalts ein weiterer Ast mit daran hängenden Knoten. Hierbei gilt der oben liegende Knoten als Elternknoten (parent) und die von ihm unmittelbar abstammenden Knoten als Kindknoten (children).



Beispiel 2: Darstellung nach „rechts“
Dieser Baum steht auch auf dem Kopf, und die Zweige wachsen nur nach rechts:

- Das oberste Element einer jeden Webseite ist html. Das Stammelement.
- Von html gehen zwei Elemente ab: head und body. Man kann also sagen, dass sowohl head als auch body Kindelemente von html und somit Geschwister sind.
- body hat nur ein Kindelement namens div#wrapper. Umgekehrt ist body das Elternelement von div#wrapper.
- div#wrapper wiederum hat vier Kinder, nämlich die div-Elemente mit den IDs kopfbereich, navibereich, textbereich und fussbereich.

Wird nun im <body> eine bestimmte Schriftgröße von 1.2em angegeben, wird dies, sofern keine direkte Änderung erfolgt auch bei den Kind-Elementen angewendet, wie z.B. div#wrapper.

Links sieht man die HTML-Struktur der Startseite *index.html* als Baum.

Der Browser gibt JavaScript die Möglichkeit, mit einer Sammlung von Methoden auf diese Baumstruktur zuzugreifen und sie zu ändern.

Beispiel:

Das HTML-id-Attribut gibt die Möglichkeit, einem HTML-Element einen eindeutigen Namen (identifier) zuzuweisen.
Beispiel:

```
<h1 id="Haupttitel">Hallo Welt!</h1>
```

Daher kann man genau diese spezielle Überschrift ansprechen und ändern. Wenn ein Element eindeutig mit einer id identifiziert wurde kann man es mit der DOM-Methode document.getElementById gesucht werden. Der Aufruf gibt das Element zurück, das zu der id passt und man kann es in eine Variable speichern. Sobald es in einer Variablen gespeichert ist kann man es mit JavaScript bearbeiten.
Um zum Beispiel den Text des Elements zurückgeben zu lassen kann man die Eigenschaft „innerHTML" diesen suchen. Danach kann man diesen z.B. ersetzten.

Übung: Überschrift mithilfe des DOM ersetzten

Erstelle eine Datei „3.2.dom.html“

```
8  <body>
9  <h1 id="Haupttitel">Hallo Digbiz Mistelbach</h1>
10 <script>
11 var titelHolen = document.getElementById("Haupttitel");
12 document.write("Der erste Titel war: " + titelHolen.innerHTML);
13 var neuerTitelText = prompt("Gib bitte einen neuen Titel ein:");
14 titelHolen.innerHTML = neuerTitelText;
15 </script>
16 </body>
```

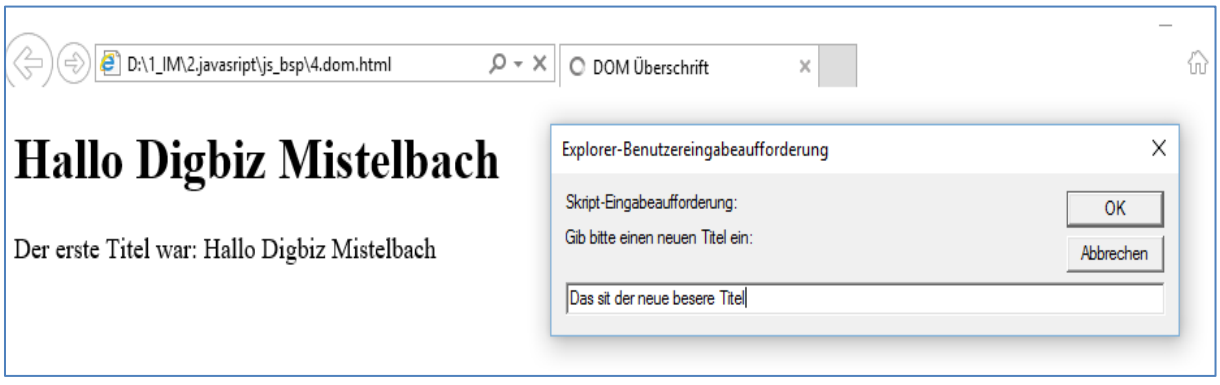
In Zeile 11 erstelle eine neue Variable „titleHolen“ und erhalte das H1-Element (mit der id=“Haupttitel“).

In der Zeile 12 gibt man einfach nur den Inhalt aus, damit man weiß, wie er gelautet hat. Das könnte man auch weglassen.

Zeile 13: Mit Hilfe eines Prompt-Dialoges fragt man den Nutzer nach einer neuen Überschrift. Den eingegebenen Text speichere in der neuen Variablen „neuerTitelText“.

In Zeile 14 wird der alte Text mit „innerHTML“ durch den neuen Text ersetzt.

Ergebnis:



2)jQuery und die Arbeit mit dem DOM-Baum

Um diese Zugriffe auf den DOM einfacher zu gestalten, nutzt man gerne ein Werkzeug namens jQuery.

jQuery ist eine JavaScript-Bibliothek, eine Werkzeugsammlung von hauptsächlich Funktionen. Zuerst muss aber diese Bibliothek in die Seite geladen und eingebunden werden, damit sie genutzt werden kann.

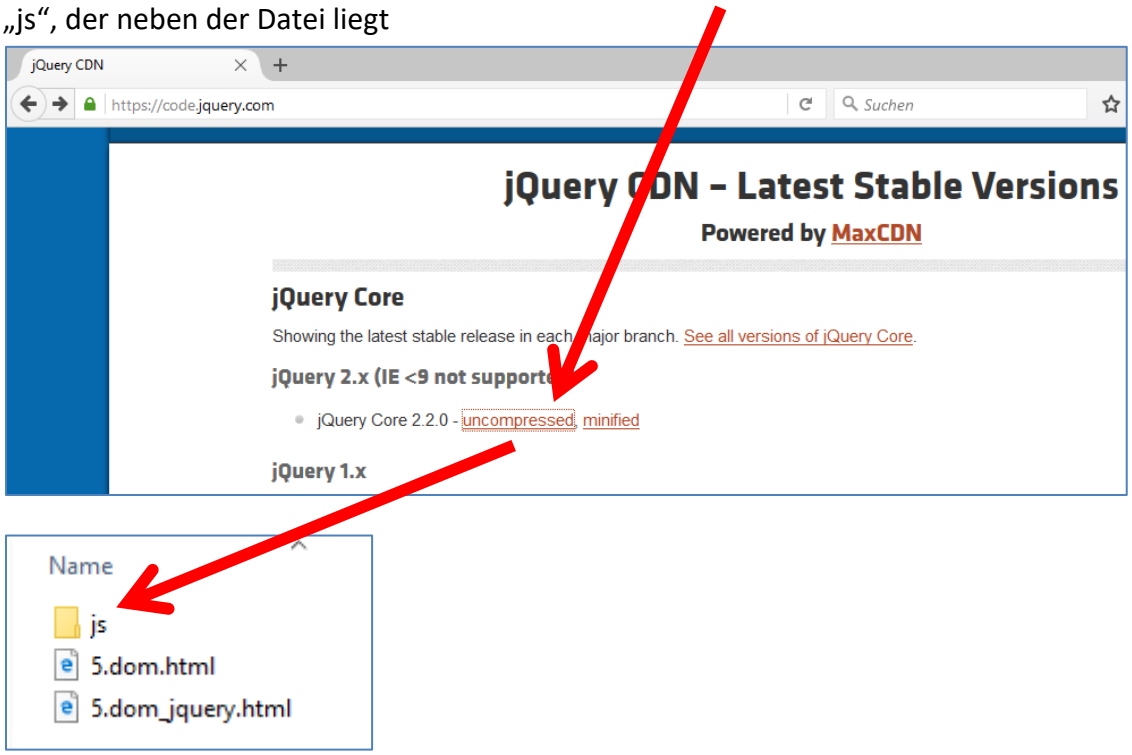
Die JavaScript-Datei für die Bibliothek ist aber ganz schön umfangreich und enthält mehr als 9.000 kompliziertes JavaScript.

jQuery in die HTML-Website laden

a)Mit einer URL (Webadresse) in die Seite einfügen

```
<script src=https://code.jquery.com/jquery-2.2.0.js></script>
```

b)Herunterladen der .js-Datei und dann kann sie auch offline in der eigenen Website zu Verfügung gestellt werden. Somit kann man jederzeit offline testen usw.
Besuche die obige Website und speichere die aktuelle Version hier 2.2.0.js in einen Ordner „js“, der neben der Datei liegt



2a)Den Überschrifttext mit jQuery ersetzten

Öffne „3.2.dom.html“ und speichere es als „3.2.dom_jquery.html“

Änderungen:

```
 8  <body>
 9  <h1 id="Haupttitel">Hallo Digbiz Mistelbach</h1>
10  <script src="https://code.jquery.com/jquery-2.2.0.js"></script>
11  <script>
12  var neuerTitelText = prompt("Gib bitte einen neuen Titel ein:");
13  $("#Haupttitel").text(neuerTitelText);
14  </script>
15  </body>
```

Oder lokal:

```
 8  <body>
 9  <h1 id="Haupttitel">Hallo Digbiz Mistelbach</h1>
10  <script src="js/jquery-2.2.0.js"></script>
11  <script>
12  var neuerTitelText = prompt("Gib bitte einen neuen Titel ein:");
13  $("#Haupttitel").text(neuerTitelText);
14  </script>
15  </body>
```

In Zeile 10 wird unterschiedlich die jQuery-Datei geladen.

In Zeile 13 wird mit **der jQuery-Funktion $** ein HTML-Element ausgewählt. **Die Funktion $** nimmt als Argument einen sogenannten „Selektorstring“ entgegen. Dieser teilt jQuery mit, welches Element oder welche Elemente aus dem DOM-Baum ausgewählt werden sollen. In diesem Fall wird „#Haupttitel“ als Argument eingegeben. Das #-Zeichen in einem Selektorstring bedeutet „ID“. Also ist der Selektorstring „#Haupttitel“ das Element mit der id Haupttitel.
Die Funktion $ gibt ein jQuery-Objekt zurück, das die ausgewählten Elemente repräsentiert. Hier gibt $(„Haupttitel“) ein jQuery-Objekt für das h1-Element zurück (das die id „Haupttitel“ besitzt).
Die Methode text in der Zeile 13 ersetzt dann den Text durch die in der Variablen „neuerTitelText“ gespeicherten Nutzereingabe.

Ergebnis:

2b)Neue Elemente mit jQuery erzeugen

jQuery kann nicht nur Elemente bearbeiten, sondern auch neue erzeugen und sie dem DOM-Baum hinzufügen. Dazu benötigt man das jQuery-Objekt „append“:

append **Die Methode „append“** konvertiert den String mithilfe der darin enthaltenen HTML-Tags in ein DOM-Element und fügt das neue Element hinter dem ursprünglichen an.
Somit kann die Website dynamisch verändert werden.

Beispiel:

$(„body“).append(„ Das ist ein neuer Absatz ganz unten “);

Im ersten Teil wählt man mit der Funktion $ mit dem Selektorstring „body“ den body-Bereich des HTML-Dokuments aus. Der Selektorstring muss keine id sein, Der Code „$(„body“) wählt genau das body-Element aus. Man könnte aber auch z.B. mit „$(„p“)“ alle p-Elemente auswählen.

Als Nächstes ruft man die Methode „append“ für das von $(„body“) zurückgegebene Objekt auf. Der nun an „append“ übergebene String wird in ein DOM-Element umgewandelt und innerhalb des body-Elements direkt vor dem schließenden Tag eingefügt.

Beispiel:

Arbeite am Dokument „3.2.dom_jquery.html“ weiter und speichere es als „3.2.dom_jquery_append.html“.
Füge vor dem <script>-Ende folgenden Code ein:

$("body").append(" Das ist ein neuer Absatz ganz unten. ");

```
11  <script>
12  var neuerTitelText = prompt("Gib bitte einen neuen Titel ein:");
13  $("#Haupttitel").text(neuerTitelText);
14  $("body").append("<p>Das ist ein neuer Absatz ganz unten.</p>");
15  </script>
16  </body>
```

Ergebnis:

2c)append mit einer FOR-Schleife nutzen

Mit Hilfe einer for-Schleife kann man bequem mehrere Elemente mit Hilfe von „append“ hinzufügen.

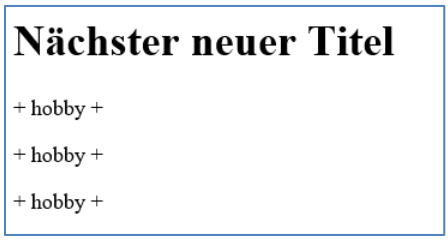
```
8   <body>
9   <h1 id="Haupttitel">Hallo Digbiz Mistelbach</h1>
10  <script src="js/jquery-2.2.0.js"></script>
11  <script>
12  var neuerTitelText = prompt("Gib bitte einen neuen Titel ein:");
13  $("#Haupttitel").text(neuerTitelText);
14  for (var i = 0; i < 3; i++) {
15      var hobby = prompt("Bitte gib eines deiner Hobbys ein!");
16      $("body").append("<p>" + hobby + "</p>");
17  }
18  </script>
19  </body>
```

Diese Schleife läuft dreimal durch. Jedes Mal erscheint ein Prompt-Dialog, der den Nutzer auffordert, ein neues Hobby einzugeben. Jedes Hobby wird dann in <p>-Tags gesetzt und der append-Methode übergeben, die das Hobby an das Ende des body-Elements anfügt.

Ergebnis:

va

Vorsicht Fehler: dabei append falsche Anführungszeichen ☺




```
14  for (var i = 0; i < 3; i++)
15      {
16      var hobby = prompt("Bitte gib eines deine
17      $("body").append("<p> + hobby + </p>");
18      }
```

3)Elemente mit jQuery animieren – fadeOut()

Um ein Element langsam animiert auszublenden, kann man die **Methode „fadeout“** verwenden.

$(„h1“).fadeOut(3000);

Mit der Funktion $ wählt man alle h1-Elemente aus. Hier im Beispiel gibt es aber nur ein h1-Element, nämlich das mit „Hallo Digbiz Mistelbach“. Dieses wird als jQuery-Objekt ausgewählt. Für dieses Objekt ruft man .fadeOut(3000) auf, um die Überschrift innerhalb von 3 Sekunden komplett auszublenden.

Die Einheit für das Argument von fadeOut sind Millisekunden oder Tausendstelsekunden. Also sind 3 Sekunden genau 3000.

Übung:
Arbeite am oberen HTML-Dokument weiter und speichere es unter „3.2.dom_jquery_fadeout.html“.
Ersetzte den kompletten zweiten Skriptteil durch die Methode wie oben

```
 8  <body>
 9  <h1 id="Haupttitel">Hallo Digbiz Mistelbach</h1>
10  <script src="js/jquery-2.2.0.js"></script>
11  <script>
12  $("h1").fadeOut(3000);
13  </script>
14  </body>
15  </html>
```

3a)jQuery-Animationen verketten

Man kann mehrere Animationen desselben Elements verketten. Dazu kann man mehrere Methoden hintereinander schreiben, wenn der Punkt dazwischen gesetzt wird.

Beispiel: Ändern des Textes und dann ausblenden

$("h1").text("Das wird gleich ausgeblendet werden.").fadeOut(3000);

```
10  <script src="js/jquery-2.2.0.js"></script>
11  <script>
12  $("h1").text("Das wird gleich ausgeblendet werden.").fadeOut(3000);
13  </script>
14  </body>
```

fadeIn()

Die „fadeIn()“-Animation blendet ein unsichtbares Element wieder ein. jQuery ist schlau genug zu wissen, dass zwei verkettete Animationen vermutlich nacheinander ablaufen werden.

$(„h1“).fadeIn(4000);

Übung:
Die <h1> soll zuerst ausgeblendet werden und dann 4 Sekunden lang eingeblendet werden. Schreibe den Code dafür.

slideUp() und slideDown()

Die Methode „slideUp“ lässt ein Element verschwinden, indem es nach oben gleitet. Mit „slideDown“ taucht es wieder auf, indem es nach unten gleitet.

Fragen:

- Was ist ein Dokumenten-Stammbaum (DOM)? Wie werden hier Verwandtschaften (Vererbung) sichtbar?
  (Stellt die Website dar als klare Struktur. Er ist die Form eines auf den Kopf gestellten Baumes. Das Wurzelelement ist <html>. Aus dem DOM werden Verwandtschaften ersichtlich. Fast jedes Element kann Kind- und Elternelement sein. Elternelemente sind näher zur Wurzel, Kind-Elemente sind näher zu den Ästen draußen.))

Quellen:

Nick Morgan, in JavaScript kinderleicht, 2015, dpunkt-Verlag, Heidelberg, S. 133-141