Die Sprache HTML im Detail

Julian Schlee
1821112 | WEB - 3IB WS19/20

Inhaltsverzeichnis

1 Einleitung 2

1.1 Die Entstehung von HTML 2

1.2 Was ist HTML? 3

1.3 HTML-Elemente 3

2 Grundlegende Steuerelemente 5

2.1 Die Grundstruktur eines HTML-Dokuments 5

2.1.1 Die Angabe des Dokumenttyps 5

2.1.2 Der Aufbau der Grundstruktur 6

2.1.3 Das head- und body-Element 6

2.1.4 Das komplette Grundgerüst 7

2.2 Die Strukturierung des Seiteninhalts 7

2.2.1 Strukturierung von Texten 7

2.2.2 Listen 8

2.2.3 Bilder 10

2.2.4 Referenzen 11

2.2.5 Tabellen 13

3 Formulare und Formulardaten 16

3.1 HTML-Formular-Elemente 16

3.2 Die Datenübertragung 17

3.3 Weitere nützliche Formular-Elemente 18

4 Das Dokument-Object-Model 21

4.1 Der Aufbau eines DOM‘s 21

4.2 Der Nutzen eines DOM‘s 22

5 Neuerungen in HTML5 24

5.1 Neue Möglichkeiten der Seitenstrukturierung 24

5.2 Multimedia-Elemente 25

A. Lösungen der Aufgaben zur Selbstüberprüfung 27

B. Literaturverzeichnis 30

C. Abbildungsverzeichnis 33

D. Codeverzeichnis 34

1. Einleitung

In diesem Kapitel wird zu Beginn kurz die Geschichte der Entstehung der Sprache HTML erzählt, um dem Leser einen ersten Einblick in den Nutzen der Sprache HTML zu geben, und ihm zu verdeutlichen welchen Einfluss die Sprache auf unser heutiges World-Wide-Web hat.

Anschließend wird der Begriff „HTML“ sowie die grundlegende Funktion der Sprache erläutert.

Nach dem durchlesen dieses Kapitels sollten Sie verstanden haben:

- wofür die Sprache HTML genutzt wird,
- welche Bedeutung sie in unserem Zeitalter hat,
- warum HTML essenziell für die Erstellung von Websites ist und
- was HTML-Elemente sind und welchen Zweck sie erfüllen

1.1 Die Entstehung von HTML

Das Konzept der Sprache HTML wurde 1989 von Sir Tim Berners Lee entwickelt. Er arbeitete damals an einer Forschungseinrichtung, der Europäischen Organisation für Kernforschung(CERN), in der Schweiz. Dort begann er 1989 an einem Projekt zu arbeiten, welches den Ursprung des heutigen World-Wide-Webs darstellt. Das Ziel dieses Projektes war, die Dokumente der verschiedenen Forschungsinstitute elektronisch zugänglich zu machen, um den weltweiten Austausch von Informationen zwischen den Wissenschaftlern zu erleichtern. (vgl. [RL, Kapitel 1])

Dazu entwickelte er 1990 den ersten Webserver (CERN httpd) um die Dokumente veröffentlichen zu können, und den ersten Browser (WorldWideWeb) um die Dokumente abrufen zu können, sowie in der Folgezeit die URI und das Transferprotokoll HTTP. (vgl. [Be19])

Damit diese Dokumente von einem Browser in einem gewünschten Format interpretiert und dargestellt werden können, suchte er nach einem Weg die Dokumente strukturieren zu können, und durch sogenannten Hypertext, direkte Referenzen auf andere Dokumente setzten zu können. Genau an diesem Punkt kam der Ansatz für die Textauszeichnungssprache HTML ins Spiel. (vgl. [RL, Kapitel 1])

Durch kontinuierliche Updates der Sprache schuf er ein Werkzeug welches dem Nutzer ermöglicht sehr einfach die Grundstruktur einer Website festzulegen.

Da heutzutage nahezu jede Website auf einer HTML-Grundstruktur basiert, schuf er damals gleichzeitig auch den weltweiten Standard der heutigen Webentwicklung.

Heutzutage ist jeder gängige Browser in der Lage einen HTML-Quellcode zu interpretieren.

1.2 Was ist HTML?

Der Begriff HTML steht für Hypertext Markup Language.

Die Sprache HTML ist ein Dialekt der Metasprache SGML(Standard Generalized Markup Language), welche schon „ab 1960 Texte standardisieren und für den Computer lesbar machen sollte“. (vgl. [Se18])

Wie der Begriff Markup-Language bereits aussagt, handelt es sich bei HTML und SGML nicht um klassische Programmiersprachen wie etwa Java oder C++, sondern um Auszeichnungssprachen. Dies bedeutet es gibt keine Funktionen, Bedingungen oder Variablen.

HTML wird also benutzt, um bestimmte Teile eines Dokuments auszuzeichnen, und dadurch dem Browser zu sagen wie er diese zu interpretieren und darzustellen hat. Diese Teile werden Elemente der Website, bzw. des Codes genannt.

Auf diese Weise lässt sich die grundlegende semantische Struktur einer Website festlegen.

Allerdings ist es durch HTML nicht möglich, weitgehende Einstellungen an der visuellen Darstellung der Website festzulegen, oder dynamische Funktionen einzubauen wie z.B. Drop-Down-Menüs. Dafür werden weitere Dateien in den HTML-Quellcode eingebunden, welche durch andere Sprachen wie CSS oder JavaScript erstellt werden.

> „HTML beschreibt nicht, wie ein Element aussieht oder wo es platziert ist, HTML beschreibt, was ein Element ist – eine Überschrift, eine Liste, ein Bild, eine Tabelle.
>
> [Sc08 , S43]

Ein HTML-Quellcode kann in einem üblichen Text-Editor verfasst und bearbeitet werden, und benötigt keinen speziellen Editor oder IDE. Ebenso kann jede HTML-Datei einfach durch einen Browser geöffnet werden, in welchem dann die Darstellung des Codes erscheint.

1.3 HTML-Elemente

Jeder HTML-Quellcode besteht aus mehreren einzelnen Elementen, welche jeweils einen Teil einer Website repräsentieren, wie z.B. ein Textabschnitt, ein Bild oder auch die Navigationsleiste.

Jedes Element wird durch ein bestimmtes Tag gekennzeichnet. Dies ist nötig, damit der Browser weiß um welche Art von Inhalt es sich handelt, und wie er diesen zu interpretieren hat.

Der Aufbau eines HTML-Elements hat die folgende Struktur:

```
<Tag-Name> Inhalt des Elements </Tag-Name>
```

Code 1.1 - Der Aufbau eines HTML-Elements

Durch das Setzen eines Start- und eines End-Tags wird der Beginn und das Ende des Inhalts gekennzeichnet. Diese Tags werden immer in spitzen Klammern **< >** eingeschlossen und das End-Tag wird durch ein **/** vor dem Tag-Namen eingeleitet. Dadurch wird auch gewährleistet, dass immer nur der Inhalt und nie die Tags selbst dargestellt werden.

Diese Elemente können auch ineinander verschachtelt werden, dazu müssen aber bestimmte Regeln eingehalten werden. Eine Navigationsleiste ist beispielsweise oft aus einer Liste und mehreren

Listeneinträgen, welche die Buttons darstellen, zusammengesetzt. Dabei wird jeder Listeneintrag als einzelnes Element innerhalb eines Listen-Elements definiert.

Hierbei ist wichtig zu beachten, dass Start- und End-Tags immer ebenentreu verschachtelt sind. Dies bedeutet, dass nach dem Öffnen eines Elementes welches innerhalb eines anderen definiert wird, kein End-Tag des Elternelementes folgen darf, bevor das Tag wieder geschlossen wurde.

Um die Elemente genauer spezifizieren zu können, beinhalten Elemente auch verschiedene Attribute.

Zusammenfassung

In diesem Kapitel wurde ihnen kurz erklärt wie die Sprache HTML entstanden ist und was die ursprüngliche Idee dahinter war. Dabei sollte ihnen auch die Bedeutung der Sprache in der heutigen Zeit nahegelegt werden.

Im Anschluss wurde noch der Begriff „HTML“ sowie die Funktion der Sprache erläutert, und noch ein kurzer Ausblick auf die Grenzen des mit HTML Möglichen gewährt.

Zuletzt wurde ihnen erklärt wie HTML-Elemente aufgebaut sind, und wozu diese dienen.

Aufgaben zur Selbstüberprüfung

Aufgabe 1.1:

Wozu wird die Sprache HTML benutzt?

Aufgabe 1.2:

Warum ist die Sprache HTML so wichtig für unser World-Wide-Web?

Aufgabe 1.3:

Wo liegen die Grenzen von HTML?

Aufgabe 1.4:

Was sind HTML-Elemente, welchen Zweck haben Sie und wie sind diese in einem Quellcode aufgebaut?

2 Grundlegende Steuerelemente

In diesem Kapitel werden ihnen zu Beginn die Basiselemente der Sprache vorgestellt, welche benötigt werden, um die Grundstruktur eines HTML-Dokuments zu bilden.

Anschließend werden ihnen die wichtigen Steuerelemente vorgestellt, welche zur Strukturierung des Seiteninhalts genutzt werden können.

Nach dem lesen dieses Kapitels sollten Sie:

- Den Aufbau eines HTML-Dokuments verstanden haben,
- die Möglichkeiten der Strukturierung des Seiteninhalts kennengelernt haben,
- und in der Lage sein ein eigenes, gültiges HTML-Dokument zu verfassen.

2.1 Die Grundstruktur eines HTML-Dokuments

Ein gültiges HTML-Dokument muss den vom World-Wide-Web-Consortium (W3C) festgelegten Standard erfüllen, um von Browsern richtig interpretiert werden zu können. Dazu muss der Quellcode alle Basis-Elemente der Sprache HTML enthalten, sowie die Angabe eines gültigen Dokumenttyps.

2.1.1 Die Angabe des Dokumenttyps

Da es heutzutage viele verschiedene Versionen der Sprache HTML gibt muss zu Beginn des Dokuments immer der Dokumenttyp angegeben werden. Dieser beschreibt die verwendete Version der Sprache der HTML, mit welcher der Quellcode geschrieben wurde.

Seit der Version HTML5 wurde die Angabe des Dokumenttyps deutlich vereinfacht und geschieht seitdem wie folgt:

```
<!DOCTYPE HTML>
```

Code 2.1 – Dokumenttypdeklaration in HTML5

Wichtig hierbei ist, dass diese Angabe immer in der **ersten Zeile** des Dokuments stehen muss, damit der Browser den nachfolgenden Code richtig interpretieren kann.

In der Version HTML4.01 war dies noch umständlicher und geschah folgendermaßen:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTsD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

Code 2.2 - Dokumenttypdeklaration in HTML4.01

2.1.2 Das html-Wurzelelement

Des Weiteren muss ein gültiges HTML-Dokuments alle Basis-Elemente der Sprache HTML enthalten.

Diese sind ineinander verschachtelt und beginnen mit dem **<html>** Tag. Dieses Element umschließt das gesamte Dokument und dient als Container für weitere Elemente die innerhalb davon definiert werden. Deswegen wird es oft auch als Wurzelelement des Dokuments bezeichnet.

Mittels **lang**-Attributes, kann dazu noch die Sprache des Seiteninhaltes angegeben werden. Attribute werden in HTML immer in das Start-Tag geschrieben, und meistens durch ein Keyword-Value-Paar definiert.

```
<html lang="de"> … </html>
```

Code 2.3 - Das html-Element mit einem lang-Attribut

In diesem Beispiel steht **"de"** für Deutsch, aber weitere Angaben wie z.B. **"en"** für Englisch sind auch möglich.

2.1.3 Das head- und body-Element

Das **head**-Element dient als Container für Metainformationen. In ihm werden Informationen über den Inhalt der Website angegeben, wie z.B. der Titel der Seite oder der verwendete Zeichensatz. Der Titel der Website wird durch das **title**-Element angegeben und ist ebenfalls ein Pflichtelement.

Die Angabe des Titels geschieht über das **title**-Element innerhalb des **head**-Elements.

```
<head>
      <title>Seitentitel</title>
</head>
```

Code 2.4 – Ein title-Element innerhalb des head-Elements

Der angegebene Title erscheint dadurch im Browser als Name des Tabs.

Über das Element **<style>** ist es möglich, mittels der Sprache CSS, innerhalb des HTML-Quellcodes bestimmte Angaben über das Design der Seite zu definieren. Um den Inhalt von dem Design zu trennen, sollte aber lieber eine externe CSS-Datei über das **<link>**-Element eingebunden werden.

Für die Angabe weiterer Meta-Informationen wird das Tag **<meta>** verwendet.

Als Beispiel die Angabe des Zeichensatzes durch das Attribut „**charset**“:

```
<meta charset="UTF-8">
```

Code 2 5 - Angabe des Zeichensatzes mit dem charset-Attribut innerhalb eines meta-Elements

Diese Angabe sollte stets getätigt werden um die korrekte Darstellung von Textzeichen (vor allem Umlauten) zu gewährleisten.

Weitere Informationen, wie z.B. der Autor der Website, eine Beschreibung der Seite oder zugehörige Keywords können über die Attribute **name** und **content** angegeben werden.

```
<meta name="author" content="author-name">

<meta name="keywords" content="HTML, CSS, WEB">
```

Code 2.6 - Angabe des Autors und von Keywords mit content- und name-Attributen innerhalb eines meta-Elements

Innerhalb des **body**-Elements wird der eigentliche (sichtbare) Inhalt der Seite definiert.

2.1.4 Das komplette Grundgerüst

Nimmt man nun all diese Pflichtelemente zusammen enthält man das Grundgerüst eines HTML-Dokuments welches dem Standard des W3C entspricht.

Dieses wird folgendermaßen notiert:

```
<!DOCTYPE html>
<html>

  <head>
    <title> Titel der Website </title>
    *ggf. noch Metadaten oder Styles
  </head>

  <body>
    *Inhalt der Website
  </body>

</html>
```

Abbildung 2.1 - Das HTML Grundgerüst

2.2 Die Strukturierung des Seiteninhalts

Nach dem das Grundgerüst des Quellcodes gestellt ist, kann der eigentliche Inhalt der Website in das **body**-Element geschrieben werden.

HTML bietet hierfür viele Tags an, mit denen z.B. einzelne Bereiche eines Textes ausgezeichnet und dadurch das Dokument strukturiert werden kann.

2.2.1 Strukturierung von Texten

Sie können beispielsweise damit anfangen, ihrer Website eine Überschrift zu geben.

Dafür stellt HTML die Elemente **<h1>** – **<h6>** bereit, welche in aufsteigender Nummerierung immer kleiner werden. Dadurch können verschiedene Ebenen eines Textes durch verschiedene Arten von Überschriften repräsentiert werden.

Um einen einfachen Text auf die Website einzufügen, wird das **<p>**-Element (*paragraph*) benutzt. Im Allgemeinen ist es üblich, einzelne Abschnitte eines Textes in diesen Tags zu gliedern.

Ein Text oder bestimmte Teile davon, können auch durch die Tags **** , **** und **<small>** hervorgehoben bzw. verkleinert werden. Alternativ können über das ****-Element, Teile eines Textes ausgezeichnet und später bearbeitet werden.

Zitate können durch das **<q>**-Tag markiert werden, und werden in dadurch in Anführungszeichen dargestellt.

Möchte man innerhalb eines Paragrafen einen Zeilenumbruch platzieren, kann man dies über das **
**-Tag(*line break*) tun.

Das **<hr>**-Element(*horizontal ruler*) stellt auf der Website eine horizontale Linie dar. Dies wird häufig benutzt, um thematische Brüche in einem Text zu signalisieren.

Hierbei ist zu beachten das dieses Element (**br** & **hr**) anstatt aus Start- und End-Tag, nur aus einem einzelnen Tag besteht und keinen Inhalt haben. Dies nennt sich *Standalone-Tag.*

Als Beispiel finden Sie hier einen HTML-Quellcode und dessen Darstellung in einem Browser:

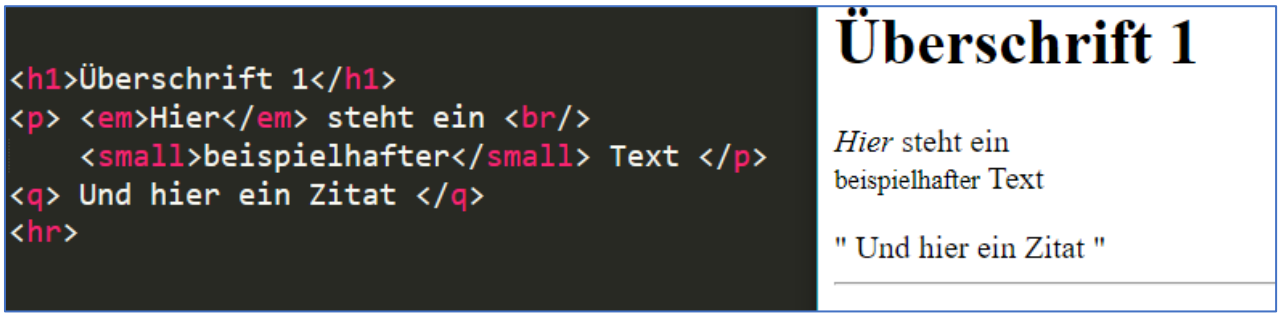

Abbildung 2.2 – Eine beispielshafte Textstrukturierung

2.2.2 Listen

Für Listen und Aufzählungen, stellt HTML drei verschiedene Varianten zur Verfügung, wobei alle Varianten unterschiedliche Darstellungen haben.

Dabei wird zwischen einer ungeordneten Liste, einer geordneten/nummerierten Liste sowie einer Definitionsliste unterschieden.

Der Aufbau der geordneten und ungeordneten Liste ist exakt ist derselbe, lediglich die Definitionsliste unterscheidet sich diesbezüglich.

Die ungeordnete Liste (ul)

Bei einer ungeordneten Liste handelt es sich um eine einfache Aufzählung ohne Nummerierung. Dies bedeutet, dass die einzelnen Listeneinträge lediglich durch ein Aufzählungszeichen eingeleitet werden.

Um eine ungeordnete Liste zu erstellen verwendet man das **<ul>**-Tag. (*unordered list*)

Innerhalb dieses Tags können durch das **<li>**-Tag einzelne Listeneinträge definiert werden.

Standardgemäß wird das Aufzählungszeichen durch einen ausgefüllten Kreis repräsentiert, aber durch das **`style`**-Attribut innerhalb des **`<ul>`**-Tags ist es durch eine Anweisung in der Style-Sprache CSS möglich, die Darstellung der Aufzählungszeichen zu verändern.

Hier ein Beispiel einer ungeordneten Liste mit **`style`**-Attribut, und ihre Darstellung:

```
<ul style="list-style-type: square">
    <li>Apfel</li>
    <li>Birne</li>
    <li>Erdbeere</li>
    <li>Zitrone</li>
</ul>
```

- Apfel
- Birne
- Erdbeere
- Zitrone

Abbildung 2.3 - Eine ungeordnete Liste mit style-Attribut

Die geordnete Liste (ol)

Bei der geordneten/nummerierten Liste handelt es sich um eine Zählliste, welche ihren Einträgen automatisch eine Ziffer zuweist und diese fortlaufend nummeriert.

Wie bereits erwähnt hat die geordnete Liste fast denselben Aufbau wie die ungeordnete Liste, lediglich die Erstellung der Liste wird hier durch das **`<ol>`**-Tag (*ordered list*) eingeleitet.

Standardgemäß wird der erste Eintrag mit der Ziffer 1 versehen. Dies kann aber durch das **`start`**-Attribut innerhalb des **`<ol>`**-Tags geändert werden. Auch die Nummerierung einzelner Listeneinträge kann durch das **`value`**-Attribut angepasst werden. Darauffolgende Einträge setzen dadurch aber automatisch bei diesem Wert fort.

Möchte man anstatt Ziffern lieber Buchstaben oder Römische Ziffern verwenden, ist dies durch das **`type`**-Attribut möglich. Der Wert **`„1“`** repräsentiert hierbei die Ziffern, **`„A“`** bzw. **`"a"`** repräsentieren Groß-/Kleinbuchstaben und **`„I“`** bzw. “**`i“`** repräsentieren große- bzw. kleine römische Ziffern.

Hier ein Beispiel einer geordneten Liste mit **`type`**- und **`start`**-Attribut, und ihre Darstellung:

```
<ol type="I" start="10">
    <li>Apfel</li>
    <li>Birne</li>
    <li value="20">Erdbeere</li>
    <li>Zitrone</li>
</ol>
```

X. Apfel
XI. Birne
XX. Erdbeere
XXI. Zitrone

Abbildung 2.3 – Eine geordnete Liste mit type- und start-Attribut

Die Definitionsliste (dl)

Bei der Definitionsliste handelt es sich um eine Liste bei der jedem Eintrag eine Beschreibung zugewiesen werden kann. Diese eignet sich besonders gut, um eine Liste von Begriffen zu definieren.

Eine Definitionsliste wird mit dem **`<dl>`**-Tag eingeleitet. Dazwischen werden die einzelnen Definitionslisten-Terme durch **`<dt>-`**Tags definiert. Dazwischen wiederum wird dann die

Definitions-Beschreibung(*definition description*) durch das **<dd>**-Tag definiert. Ein Term kann hierbei mehrere Beschreibungen haben.

Nachfolgend ein Beispiel einer Definitionsliste, und ihre Darstellung in einem Browser:

```
<dl>
    <dt> Apfel
        <dd>rundliche, fest-fleischige, aromatisch schmeckende <br>
        Frucht mit Kerngehäuse; Frucht des Apfelbaums</dd>
    </dt>

    <dt> Birne
        <dd>meist eirunde, sich zum Stiel hin verjüngende grüngelbe <br>
        oder bräunliche Frucht des Birnbaums mit saftigem <br>
        Fruchtfleisch und Kerngehäuse</dd>
    </dt>
</dl>

<br><br>
```

Apfel
rundliche, fest-fleischige, aromatisch schmeckende Frucht mit Kerngehäuse; Frucht des Apfelbaums
Birne
meist eirunde, sich zum Stiel hin verjüngende grüngelbe oder bräunliche Frucht des Birnbaums mit saftigem Fruchtfleisch und Kerngehäuse

Abbildung 2.4 - Der Quellcode einer Definitionsliste und ihre Darstellung

2.2.3 Bilder und Figuren

Über das **img**-Element ist es möglich, Bilder oder animierte GIF-Dateien in eine Website einzubinden.

Dieses Element wird als Standalone-Tag realisiert und wird lediglich über seine Attribute definiert.

Die Angabe des **src**- und des **alt**-Attributs sind hierbei notwendig, um das Bild darstellen zu können.

Das **alt**-Attribut (*alternative*) bekommt als Wert einen Text übergeben, welcher angezeigt werden wird, falls das Bild nicht korrekt geladen werden konnte.

Über das **src**-Attribut (*source*) muss die Quelle des Bildes definiert werden. Dabei wird dem Attribut entweder eine URL oder eine relative Pfadangabe zum Ablageort des HTML-Dokuments zugewiesen. Befindet sich das Bild in demselben Ordner wie die HTML-Datei in die es eingebunden wird, so kann dem Attribut als Wert einfach der Name des Bildes als Wert zugewiesen werden.

Hier als kleines Beispiel eine relative Pfadangabe und eine URL, als Wert des **src**-Attributes:

```
<img src="bild.jpg" alt="Alternativtext">

<img src="https://www.website.com/bild.jpg" alt="Alternativtext">
```

Code 2.7 – Die möglichen Pfadangaben durch ein src-Attribut des img-Elements.

Weitere Attribute, welche genutzt werden können, um die Maße des Bildes anzugeben, sind **width** und **height**. Seit der Version HTML5 wird als Wert nur noch die Angaben in Pixeln zugelassen. Es können keine Maßeinheiten mehr angegeben werden. Um die Ladezeit der Website zu verbessern wird stets empfohlen diese Werte anzugeben.

Möchte man dem Bild eine Beschreibung hinzufügen, ist dies möglich in dem man das **img**-Element innerhalb eines **figure**-Elements definiert. Dazu wird neben dem img-Element noch ein **figcaption**-Element definiert, welches als Inhalt den Beschreibungstext des Bildes bekommt.

Als Beispiel ein Quellcode eines eingebundenen Bildes mit einem Beschreibungstext, und die dazugehörige Darstellung in einem Browser:

```
<figure>
    <img src="bild.png" alt="Alternativtext"
         width="200" height="200"           >
    <figcaption>Die Beschreibung des Bildes</figcaption>
</figure>
```

Die Beschreibung des Bildes

Abbildung 2.5 – Einbindung eines Bildes, mit einem Beschreibungstext und Maßangaben [Re19]

2.2.4 Referenzen

Bei der Entwicklung von HTML war es ein grundlegender Wunsch von Sir Tim Berners Lee, dass es möglich sein sollte, innerhalb eines Dokumentes direkte Referenzen auf ein anderes Dokument setzen zu können, und diese mit nur einem Klick erreichen zu können. Oft nennt man diesen Vorgang auch das „linken“ eines anderen Dokuments. Dies ist ebenfalls die grundlegende Methode, um die Navigation von einer Seite zu einer anderen zu ermöglichen.

Das anchor-Tag

Möchte man in HTML eine Referenz zu etwas setzen, geschieht dies immer über das anchor-Element, welches mit einem **<a>**-Tag eingeleitet wird, und durch ****-Tag geschlossen wird.

Innerhalb dieses Tag muss über das **href**-Attribut die Quelle des zu verlinkenden Dokuments angegeben werden. Der Begriff **href** steht hierbei für *Hypertext-Reference*. Genau wie bei der Verknüpfung von Bildern, geschieht dies entweder über eine URL die zu diesem Dokument verweist, oder durch eine relative Pfadangabe zu dem Ablageort des HTML-Dokuments.

```
<a href="einDokument.html" …. > Ich bin ein Link </a>

<a href="https://www.eine-website.de/einDokument.html" … >
Ich bin ein Link </a>
```

Code 2.8 – Die möglichen Pfadangaben beim Setzen einer Referenz durch das href-Attribut

Zwischen diesen Tags wird der Inhalt dieses Elements platziert. Dabei kann es sich um einen einfachen Text, als auch um Bilder oder ähnliches handeln. Wird ein Text angegeben, so wird dieser, innerhalb des Browsers, unterstrichen und in blauer Farbe dargestellt. Dies dient dazu dem Nutzer zu signalisieren, dass es sich um einen Link und nicht um einen normalen Text handelt.

Fährt der Nutzer mit seinem Cursor über diesen Link oder das Bild, ändern sich das Cursorzeichen zu einer Hand. Durch einen Klick auf diesen Link oder das Bild, wird dann die referenzierte Seite geöffnet.

Das target-Attribut

Mittels des **`target`**-Attributs wird festgelegt, wie genau das referenzierte Dokument geöffnet wird.

Der Wert **`_blank`** deutet hierbei darauf hin, dass das Dokument in einem neuen Browser-Tab geöffnet werden soll. Soll das Dokument in demselben Fenster geöffnet in dem der Link angeklickt wurde, wird der Wert **`_self`** benutzt. Weiter mögliche Werte sind **`_top`** und **`_parent`**. Diese öffnen das Dokument in dem obersten bzw. in dem Elternframe des aktuellen Elements.

Nachfolgend finden Sie ein Beispiel von einem Quellcode der mittels **`anchor`**-Element und **`href`**-Attribut ein anderes HTML-Dokument (main.html) referenziert. Da es sich bei dem Inhalt um eine **`figure`** mit **`img`** und **`figcaption`** handelt, und als **`target`** „**`_blank`**“ angegeben wurde, wird diese Referenz in Form eines Bildes mit einem dazugehörigen Link dargestellt. Durch einen Klick wird das referenzierte Dokument in einem neuen Browserfenster geöffnet.

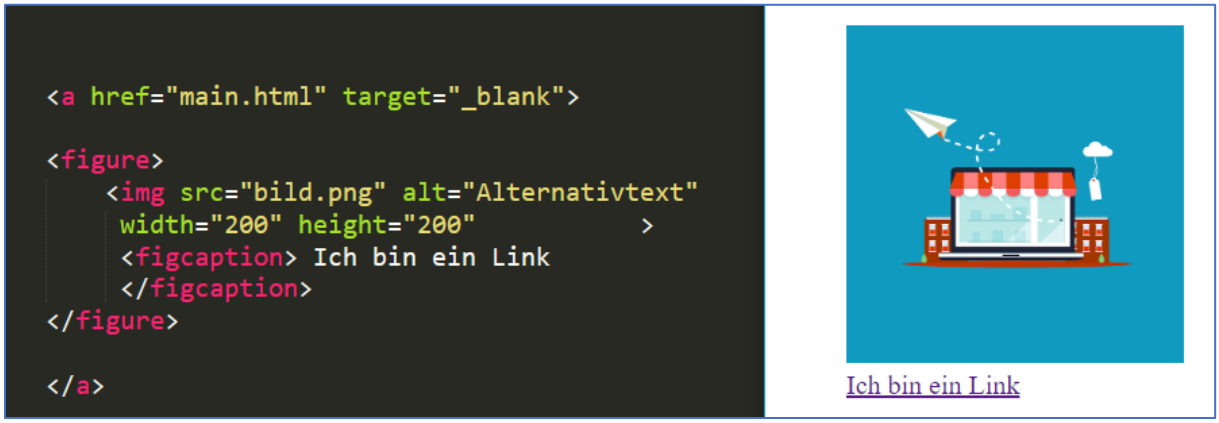

Abbildung 2.6 - Referenzierung des Dokuments „main.html“, durch ein anchor um das eingebundene Bild [Re19]

Das download-Attribut

Über das Attribut **`download,`** ist es möglich dem Nutzer das Herunterladen der referenzierten Datei zu ermöglichen. Dieses Attribut wird ebenfalls in das **`anchor`**-Tag geschrieben. Weist man diesem Attribut einen Wert zu, dann legt man dadurch einen default-Namen für die Datei fest.

Anstatt den Nutzer auf ein anderes Dokument weiterzuleiten, wird durch einen Klick auf das referenzierte Element ein Downloadfenster geöffnet. Hier kann der Nutzer wie üblich den Ablageort der Datei bestimmen und diese dort speichern.

Sprünge und id’s/div‘s

Des Weiteren ermöglicht das Anchor-Element, Verweise auf bestimmte Teile der Website erstellen zu können. Dies ermöglicht dem Nutzer durch einen Klick von einem Abschnitt der Seite zu einem anderen zu Springen. Dies kann beispielsweise langes Scrollen durch Texte vermeiden, weil dadurch ein bestimmtes Kapitel direkt angesprungen werden kann.

Dazu muss das Ziel des Sprungs durch eine frei wählbare **id** ausgezeichnet werden. Jedem HTML-Element kann eine **id** zugewiesen werden, um dieses eindeutig identifizieren zu können. Der Name besteht aus Text und darf keine Leerzeichen enthalten. Id's werden ebenfalls verwendet, um die weitere Verarbeitung durch andere Web-Sprachen zu ermöglichen.

```
<h1 id="Überschrift1"> … </h1>
```

Code 2.9 – Die Zuweisung einer id für ein h1-Element

Alternativ dazu kann auch ein **div**-Element (division) erstellt werden, welches als Container für andere HTML-Elemente dient und dadurch einen semantisch einheitlichen Bereich einer Website kennzeichnet.

```
<div id="Bereich1"> … </div>
```

Code 2.10 - Die Zuweisung einer id für ein div-Element

Innerhalb des **Anchor**-Elements kann dann als Wert des **href**-Attributs, der Name der **division** oder die **id** des betreffenden Elements eingetragen werden. Wird eine **id** verwendet werden, muss dies durch ein **#** vor dem **id**-Namen gekennzeichnet werden.

```
<a href="#Überschrift1"> Hier gelangen Sie zu Überschrift 1
</a>
```

Code 2.11 - Das Festlegen eines Sprungs, durch eintragen einer id in einem href-Attribut

2.2.5 Tabellen

Eine weitere Funktion der Sprache HTML ist das Erstellen von Tabellen für Webseiten.

Für den Aufbau einer Tabelle, sind mehrere Elemente notwendig. Das Einleiten einer Tabelle geschieht durch das **<table>-**Tag. Dieses Element dient als Container für den Tabelleninhalt.

Um Daten in die Tabelle einzufügen, wird zu Beginn, durch das **<tr>**-Tag (*table row*) eine Tabellenreihe innerhalb des **table**-Elements definiert. In dieser kann dann durch das Hinzufügen von **<th>**-Elementen(*table head*), eine bestimmte Anzahl von Spalten, sowie deren Überschrift definiert werden.

Die eigentlichen Einträge der Tabelle, werden anschließend, in einer neuen Tabellenreihe, durch das **<td>-**Tag (*table data*) definiert. Dabei wird das erste **td**-Element der ersten Spalte zugeordnet, das zweite Element der zweiten Spalte, und so weiter.

Daraus ergibt sich beispielsweise folgender Aufbau:

```
<table>
<tr>  <th> Spalte 1 </th>  <th> Spalte 2</th>  <th>Spalte 3</th> </tr>
<tr>  <td> Daten </td>  <td> Daten </td>  <td> Daten </td>  </tr>
<tr>  <td> Daten </td>  <td> Daten </td>  <td> Daten </td>  </tr>
<tr>  <td> Daten </td>  <td> Daten </td>  <td> Daten </td>  </tr>
</table>
```

Code 2.12 – Die Struktur einer Tabelle in einem HTML-Quellcode

Diese Tabelle wird in einem Browser folgendermaßen dargestellt:

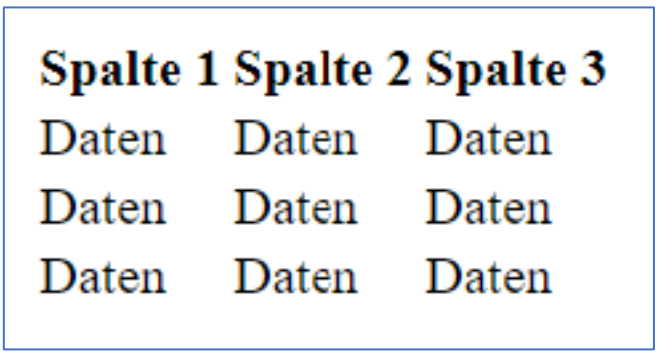

Abbildung 2.7 - Die Darstellung einer HTML-Tabelle im Browser

Durch die Verwendung eines **`caption`**-Elements kann der Tabelle auch eine Überschrift zugewiesen werden. Dieses sollte über der ersten Tabellenreihe platziert werden.

Weitere Einstellungen wie z.B. einen Rand um die Zellen der Tabelle, werden über die Sprache CSS festgelegt.

Zusammenfassung

In diesem Kapitel wurden ihnen die wichtigsten Steuerelemente der Sprache HTML vorgestellt.

Zu Beginn wurde ihnen die Grundstruktur eines HTML-Dokuments erklärt. Dabei wurde erklärt welche HTML-Elemente das Dokument mindestens besitzen muss, um den vom W3C festgelegten Standard zu erfüllen, und dadurch vom Browser richtig interpretiert werden zu können.

Anschließend wurden ihnen die wichtigsten Steuerelemente der Sprache vorgestellt, welche dazu dienen den eigentlichen Inhalt der Website zu definieren und zu strukturieren. Dabei wurde ihnen vorgestellt wie Sie Textinhalte formatieren, Bilder einbinden, Referenzen setzen, sowie die Erstellung von Listen und Tabellen.

Aufgaben zur Selbstüberprüfung

Aufgabe 2.1:

Welche Elemente werden vorausgesetzt, um die Gültigkeit eines HTML-Elements zu gewährleisten zu können?

Aufgabe 2.2:

Welche zusätzliche (optionale) Angabe sollten getätigt werden, um eine korrekte Darstellung von Texten zu gewährleisten?

Aufgabe 2.3:

Beschreiben Sie anhand eines kurzen Quellcodes die Grundstruktur eines HTML-Dokuments.

Aufgabe 2.4:

Nennen Sie mindestens 2 Elemente die benötigt werden, um einen Text strukturiert auf einer Website darzustellen. Sowie 2 Elemente, die einen Text hervorheben können.

Aufgabe 2.5:

Welche Möglichkeiten gibt es Listen zu definieren, und wie unterscheiden sich diese?

Aufgabe 2.6:

Wie wird ein Bild in eine Website eingebunden und welche Möglichkeiten gibt es dabei?

Aufgabe 2.7:

Wie werden Referenzen auf eine externe Website gesetzt? Und wie werden Sprünge innerhalb einer Website ermöglicht?

Aufgabe 2.8:

Welche Elemente stellt HTML zur Erstellung einer Tabelle bereit, und welche Funktion haben diese?

3 Formulare und Formulardaten

Um eine strukturierte Eingabe und Verarbeitung von Daten gewährleisten zu können, werden in HTML Formulare verwendet.

Um diese Formulare beliebig anpassen zu können, stellt HTML eine Reihe von Formular-Elementen bereit. Diese werden ihnen in diesem Kapitel vorgestellt.

Des Weiteren werden ihnen die Elemente und Attribute vorgestellt, die verwendet werden, um die Datenübertragung der eingegebenen Daten zu konfigurieren.

Nach dem Lesen dieses Kapitels sollten die verstanden haben, wie:

- Sie in HTML ein Formular erstellen,
- welche Elemente HTML dafür bereitstellt und
- wie Sie die Datenübertragung vorkonfigurieren.

3.1 HTML-Formular-Elemente

Die Erstellung eines Formulars beginnt mit dem Deklarieren eines **`form`**-Elements (*formular*), welches als Container für die Formular-Elemente dient. Es können aber auch die Standard-HTML-Elemente benutzt werden.

Das wichtigste dieser Formular-Elemente ist das **`input`**-Element. Dieses dient dazu, dem Nutzer ein Eingabe- bzw. Auswahlfeld bereitzustellen. Dazu hat es viele unterschiedliche Darstellungsformen, welche eine präzise Auswahl, über die gewünschten Art der Nutzereingabe ermöglichen.

Durch das **`type`**-Attribut kann diese Art von Eingabe genau festgelegt werden. Zu den unterschiedlichen Typen gehören beispielsweise simple Text-Eingabefelder (**`text`**), Ankreuzfelder (**`checkbox`**), Datum- und Farbwähler (**`date`**/**`color`**) sowie ein Datei-Upload-Button (**`file`**).

Durch weitere Werte wie **`text`** , **`number`** oder **`email`** kann festgelegt werden welche Eingaben möglich sind bzw. akzeptiert werden.

Viele dieser Typen besitzen Attribute wie **`size`**, **`maxlength`** und **`min`**/**`max`**. Durch welche die Größe des Eingabefeldes, die maximale Zeichenanzahl und eine Unter-/Obergrenzen für eingetragene Werte festgelegt werden kann.

Durch das Attribut **`required`** kann festgelegt werden welche Input-Element Pflichtfelder sind und welche nicht. Die Beschriftung dieser Pflichtfelder wird üblicherweise mit einem **`*`** gekennzeichnet.

Das nachfolgende Bild zeigt die Darstellung einiger **input-types**, und den dazugehörigen Code.

Abbildung 3.1 - Die verschiedenen Input-Typen von Formularen

Um dem Button einen Namen zu geben wurde in der Abbildung das Attribut **value** verwendet.

Mit dem Wert **„reset"** des **type**-Attributs kann ein spezieller Button zum Zurücksetzen der Eingaben definiert werden. Dieser setzt alle Eingaben im aktuellen **form**-Element zurück.

Der Wert **„submit"** hingegen wurde als default für **button**-Elemente definiert, und sorgt dafür, dass die eingegebenen Daten an eine andere Seite weitergeleitet werden, wo sie ggf. weiterverarbeitet werden.

3.2 Die Datenübertragung

Da die Datenübertragung in Form von „Schlüssel-Wert-Paaren" stattfindet (vgl. [Se19]), muss jedem **input**-Element auch ein bestimmter Schlüssel, also ein Name zugewiesen werden. Dies geschieht über das Attribut **name.** Nur so können die übertragenen Daten, auf der Empfängerseite richtig verstanden werden. Oft wird zu Verarbeitung der Daten, ein serverseitiges Skript verwendet, welcher die Daten beispielsweise in einer Datenbank abspeichert.

Um festzulegen wohin genau die Daten übertragen werden sollen, wird dem Attribut **action**, innerhalb des form-Elements, eine URL/ein Dateipfad zugewiesen. Ebenfalls wird über das Attribut **method** die entsprechende HTTP-Methode angegeben, mit der die Daten übertragen werden sollen. Üblicherweise wird hier „**post**" verwendet, da die Methode „**get**" die übergebenen Daten in der URL darstellt, was für sensible Daten nicht geschehen sollte. (vgl. [Se19])

Das nachfolgende Bild zeigt den Quellcode eines simples Kontaktformulars, welches die Werte von „Name" und „Nachricht" mittels HTTP-POST Methode an die Seite „formularVerarbeitung.php" weiterleitet. Sowie dessen Darstellung in einem Browser.

```
<form action="formularVerarbeitung.php" method="post">

    Name:<br> <br>
    <input type="text" name="Name"> <br> <br>
    Nachricht:<br> <br>
    <textarea name="Nachricht" rows="5"></textarea> <br>
    <input type="submit" value="Abschicken">
</form>
```

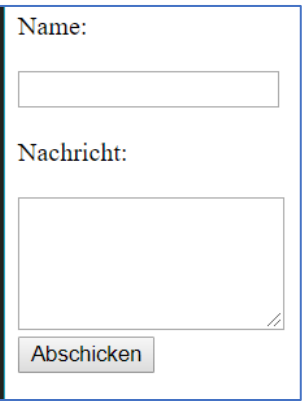

Abbildung 3.2 – Ein Kontaktformular mit Konfiguration zur Datenübertragung

3.3 Weitere nützliche Formular-Elemente

Um das Eingabefeld zu erweitern wurde in der Abbildung 3.2 ein **`textarea`**-Element verwendet, dieses verhält sich wie ein **`input`**-Element vom Typ „**`text`**“, stellt aber eine größere Eingabefläche zur Verfügung. Die Maße können über die Attribute „**`rows`**“ (Reihen) und „**`cols`**“ (Spalten) festgelegt werden.

Mittels **`select`**- und dazugehörigen **`option`**-Elementen kann eine Liste von Auswahlmöglichkeiten für ein Eingabefeld bereitgestellt werden. (siehe Seite 19, Abb. 3.3, Feld: „Geschlecht“)

Für die Beschriftung eines Input-Feldes kann auch das Element **`label`** verwendet werden. Dazu wird der jeweilige Beschreibungstext, als Inhalt in das **`label`**-Element geschrieben. Dem Attribut **`for`** muss als Wert, der Name des **`input`**-Elements auf welches es sich bezieht, zugewiesen werden. Dies stellt eine logische Verknüpfung der Elemente her, welche vor allem für die weitere Bearbeitung des Formulars, durch CSS/JavaScript oder PHP von Nutzen ist. An der Darstellung ändert sich allerdings nichts.

Ein Input-Feld mit einem zugehörigen Label hat die folgende Form:

```
<label for="Namensfeld"> Geben Sie ihren Namen ein: </label>

<input type="text" name="Namensfeld">
```

Code 3.1 - Die Deklaration eines label-Elements, welches mit einem input-Feld zusammenhängt

Des Weiteren ist es durch das Attribut **`value`** möglich, einem input-Element einen vordefinierten *default*-Wert zu vergeben, welcher verwendet wird, falls der User das Formular abschickt, ohne das Feld auszufüllen.

```
<input type="text" name="Vorname" value="Michael">
```

Code 3.2 - Das Festlegen eines default-Wertes für ein input-Element

Soll ein input-Element nicht verändert werden können, kann dies durch das Attribut **`readonly`** festgelegt werden.

In dem nachfolgenden Bild sieht man einen Quellcode eines Nutzer-Registrierung-Formulars, welches einige der bisher erwähnten Elemente beinhaltet. Sowie dessen Darstellung in einem Browser:

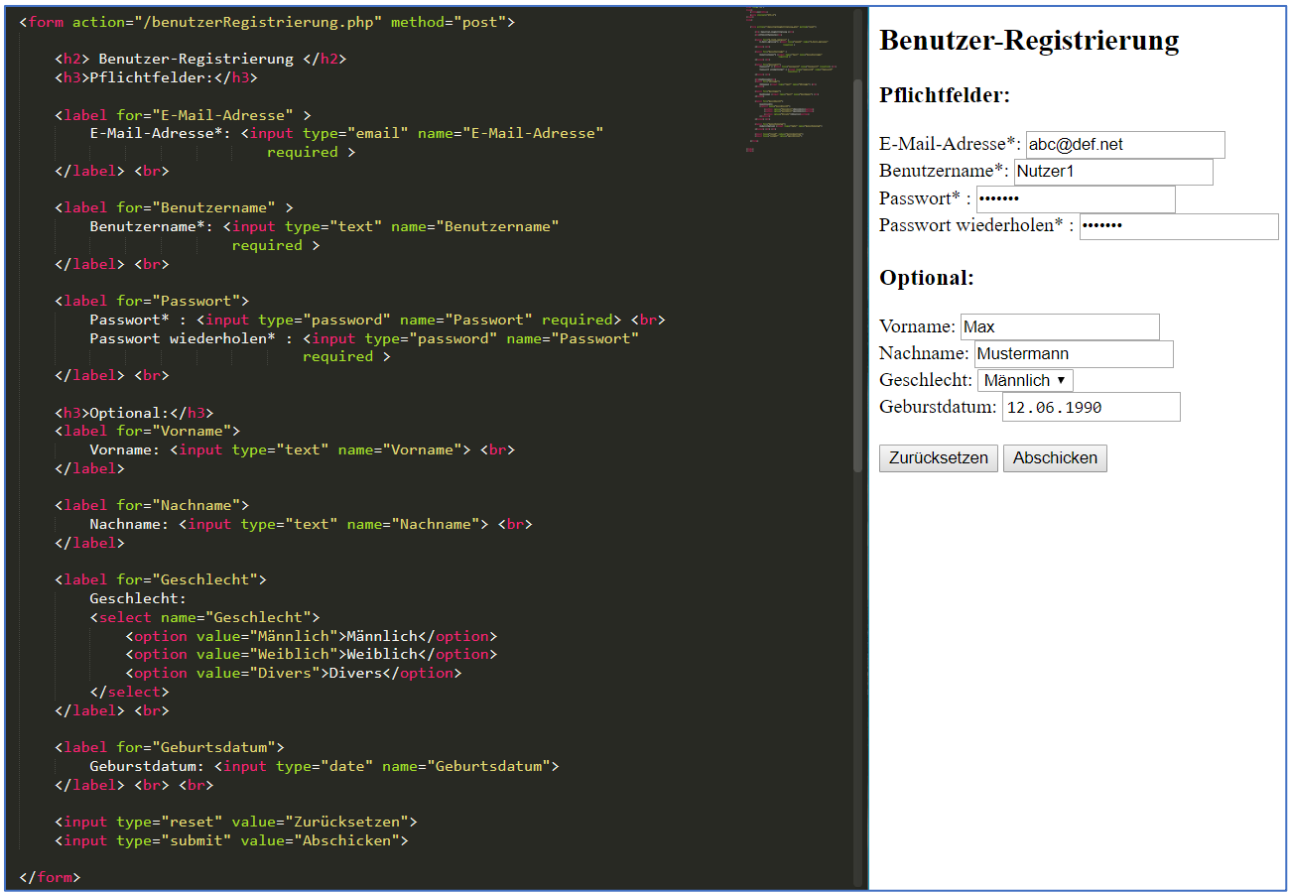

Abbildung 3.3 – Der Quellcode eines Registrierung-Formulars, sowie dessen Darstellung in einem Browser.

Über das **`button`**-Element können auch benutzerdefinierte Buttons erstellt werden. Um diesen aber eine Funktion zuordnen zu können, muss die Skriptsprache JavaScript verwendet werden. Der Funktionsaufruf geschieht über das Attribut **`onclick`**.

```
<button value="Button-Name" onclick="JavaScript-Funktion()">
```

Code 3.3 – Die Definition eines Buttons mit benutzerdefinierter JavaScript-Funktion

Zusammenfassung

In diesem Kapitel wurde ihnen vorgestellt wie mithilfe von HTML, Formulare erstellt werden können.

Dazu wurden ihnen die wichtigsten Formular-Elemente vorgestellt, mit denen Formulare individuell angepasst werden können, und Eingabefelder auf die gewünschten Eingabemöglichkeiten reduziert werden können.

Des Weiteren wurde ihnen erklärt wie Sie die Formulare so konfigurieren, dass die eingegebenen Daten auch korrekt übertragen werden können.

Aufgaben zur Selbstüberprüfung

Aufgabe 3.1:

Wie wird ein Formular definiert, und wie werden Eingabefelder innerhalb eines Formulars definiert? Nennen Sie drei Beispiele für den Typ eines Eingabefeldes.

Aufgabe 3.2:

Welche Attribute werden in Formularen vorausgesetzt, um eine Datenübertragung zu ermöglichen, und was bewirken diese?

4 Das Dokument-Objekt-Modell

In diesem Kapitel wird ihnen das Dokument-Objekt-Modell (kurz DOM) vorgestellt.

Zu Beginn wird ihnen die Darstellung, die Struktur und der Aufbau des Modells erläutert.

Anschließend wird ihnen die eigentliche Funktion des Modells beschrieben, und verdeutlicht was durch die Erstellung eines DOM`s ermöglicht wird.

Nach dem durchlesen dieses Kapitels sollten Sie verstanden haben:

- Wie das DOM aufgebaut wird und
- welchen Nutzen es im Bereich Webentwicklung hat.

4.1 Der Aufbau eines DOM´s

Durch die Eigenschaft der Sprache HTML, dass alle Elemente ineinander verschachtelt sind, lässt sich daraus eine Struktur abbilden, die sich als Dokument-Objekt-Modell bezeichnet.

Dieses Modell ist in mehrere Objekte unterteilt, welche die einzelnen Elemente des HTML-Quellcodes repräsentieren und miteinander in einer logischen Beziehung stehen können.

Ein Objekt kann dabei mehrere Kinderobjekte besitzen. Dies dient zur Repräsentation einer „Container“-Beziehung wie z.B. die einer Liste zu ihren Listeneinträge.

Auch das ganze Dokument selbst wird dabei als Document-Objekt repräsentiert, und bildet den Ausgangspunkt der gesamten Hierarchie. Von ihm aus ist jedes andere Objekt des Baumes erreichbar.

Das Dokument-Objekt besitzt als Kind ein Objekt, welches das **`<html>`**-Wurzelelements des Dokuments wiederspiegelt. Dieses wiederum besitzt das **`<head>`**- und **`<body>`**-Element als Kinder, und so weiter bis das alle Elemente des Dokuments abgebildet wurden.

Falls ein Element bestimmte Attribute besitzt, werden diese ebenfalls in dem Modell durch eine logische Beziehung zu einem Attribut-Objekt dargestellt.

Nachdem alle Elemente, ihre Attribute und die logischen Beziehungen in das Modell übertragen wurden, ergibt sich daraus eine Baumstruktur:

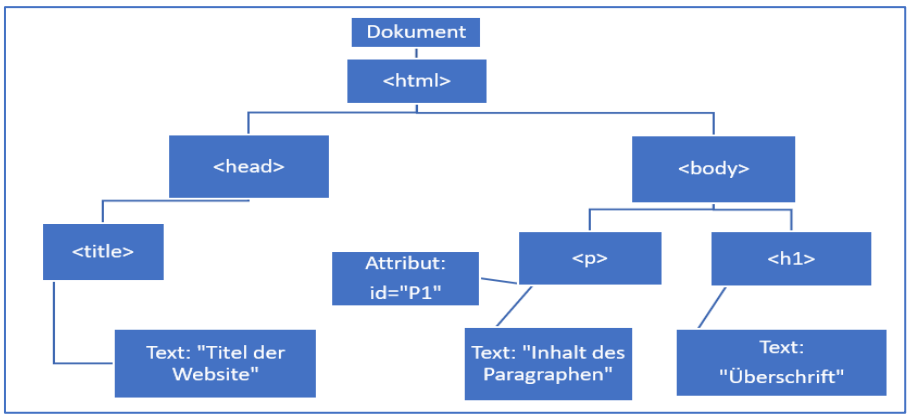

Abbildung 4.1 - Darstellung eines HTML-Dokuments in Baumstruktur; (vgl. [PH04, Kapitel 2])

4.2 Der Nutzen eines DOM`s

> „The Document Object Model (DOM) is an application programming interface (API) for valid HTML and well-formed XML documents. It defines the logical structure of documents and the way a document is accessed and manipulated.“
>
> [PH04, Kapitel 1]

Wie das Zitat bereits ausdrückt, handelt es sich bei dem DOM nicht nur um eine Abbildung der logischen Struktur eines Dokuments, sondern auch um eine Programmierschnittstelle. Diese ermöglicht einen externen Zugriff, sowie die externe Manipulation der HTML-Elemente durch eine Skript- oder Programmiersprache.

Dies bildet die Grundlage, die benötigt wird, um eine Website dynamisch gestalten zu können.

Hauptsächlich wird in diesem Fall JavaScript verwendet. Aber das DOM wurde so entworfen, dass die Realisierung auch durch andere Sprachen wie z.B. Python, C++ oder PHP möglich ist. (vgl. [W3C04])

Wird ein HTML-Dokument durch einen Browser geladen, so erstellt er ein solches Modell auf Basis dieses Dokuments und speichert das Modell in dem Arbeitsspeicher des Nutzers. (vgl. [Se19])

Auf dieser Grundlage kann z.B. ein Skript in JavaScript erstellt werden, welcher in die HTML-Datei eingebunden wird. Innerhalb dieses Skripts kann dann durch die bereitgestellten Methoden der Schnittstelle, auf jedes HTML-Elemente sowie deren Attribute zugegriffen werden.

Diese können dann durch die Skript-/Programmiersprache beliebig verändert, hinzugefügt oder gelöscht werden.

Dadurch wird ermöglicht dynamische Funktionen auf der Website bereitzustellen, welche beispielsweise die visuelle Darstellung einiger Elemente verändern oder ein Ergebnis einer Berechnung liefern.

Im Fall von JavaScript stellt das DOM-API einige Methoden bereit, um gezielt auf Elemente bestimmter Klassen oder id’s zugreifen zu können. Dadurch sollte an dieser Stelle nochmal die Bedeutung dieser Attribute verdeutlicht werden.

Zusammenfassung

In dem ersten Unterkapitel wurde ihnen beschrieben woraus ein Dokument-Object-Modell besteht und wie es aus einem HTML-Dokument abgeleitet und aufgebaut wird.

Danach wurde ihnen die Funktionsweise der DOM-Programmierschnittstelle erklärt, und verdeutlicht wie dadurch die Erstellung einer dynamischen Website ermöglicht wird.

Aufgaben zur Selbstüberprüfung

Aufgabe 4.1:

Woraus ist ein DOM aufgebaut, was wird dadurch repräsentiert und wie wird das Modell dargestellt?

Aufgabe 4.2:

Welchen Nutzen hat das DOM, und was wird dadurch ermöglicht?

Aufgabe 4.3:

Was sollte verwendet werden, um Zugriffe auf das DOM zu erleichtern?

5 Neuerungen in HTML5

In diesem Kapitel werden ihnen die wichtigsten Neuerungen der Sprache HTML vorgestellt, die durch die Version HTML5 erschienen sind.

Zu Beginn werden ihnen die neuen Elemente vorgestellt, die eingeführt wurden, um die Möglichkeiten der Seitenstrukturierung zu erweitern.

Anschließend werden ihnen die neue Multimedia-Elemente vorgestellt, welche es erlauben Grafiken zu erzeugen und Multimedia-Dateien in eine Website einzubinden.

Nach dem durchlesen dieses Kapitels sollten Sie,

- die neuen Strukturierungselemente kennengelernt haben und
- Multimedia-Elemente definieren und einbinden können.

5.1 Neue Möglichkeiten der Seitenstrukturierung

Um die Möglichkeiten der Seitenstrukturierung zu verbessern und zu modernisieren, wurden in der Version HTML5 eine Reihe neuer Elemente eingeführt.

Vor HTML5 war die einzige Möglichkeit der Unterteilung des Seiteninhalts in semantische Bereiche, die Verwendung von ** **-Elemente und der Auszeichnung dieser durch entsprechende **id**- oder **class**-Attribute. Da dies oft sehr unübersichtlich wurde, wurde in HTML5 ein neues Prinzip der Seitenstrukturierung eingeführt.

Durch das **header**-Element kann nun, innerhalb des **body**'s ein Seitenkopf definiert werden. Innerhalb diesem werden dann gewöhnlich die Elemente definiert, die konstant auf jeder Seite der Homepage, im oberen Bereich der Website zu finden sind, wie beispielsweise das Logo oder die Navigationselemente. Um den Navigationsbereich auszuzeichnen kann dazu noch das **nav**-Element verwendet werden.

Das Gegenstück dazu ist das **footer**-Element, welches Elemente im unteren Bereich der Website definiert, wie beispielsweise die Kontaktdaten einer Firma und Links zu einem Impressum oder zu der Datenschutzerklärung.

Um den Bereich des hauptsächlichen Seiteninhalts zu kennzeichnen, kann nun das **main**-Element verwendet werden. Innerhalb dieses Elements können weitere Bereiche durch die neuen **article**- und **section**-Elemente gekennzeichnet werden.

Das **article**-Element sollte verwendet werden, wenn der Inhalt auf der Website für sich alleinsteht, wie ein Blog-Post oder Nachrichtenartikel. Die **section**-Elemente hingegen, sollten beispielsweise die einzelnen Kapitel eines Artikels auszeichnen. (vgl. [Se19, Kapitel 2])

Um einen Bereich zu kennzeichnen, der seitlich auf einer Website zu finden ist, kann nun das **aside**-Element verwendet werden. Dies übernimmt allerdings nicht die Positionierung der Elemente, sondern zeichnet nur diese Gruppe von Elementen aus, um Sie beispielsweise später mit CSS platzieren zu können.

5.2 Multimedia-Elemente

Erzeugung von Grafiken

Mittels **`svg`**-Elements wurde eine Möglichkeit eingeführt, direkt im HTML-Quell eine *Scalable-Vector-Graphic* zu erstellen. Innerhalb dieses Elements kann durch weitere Elemente wie beispielsweise **`circle`** oder **`rect`** (*rectangle*) eine Grafik erstellt werden. Dazu stehen noch mehrere Attribute zur Verfügung, mit denen Größe und Farbe angepasst werden können.

Das **`canvas`**-Element markiert einen rechteckigen Bereich auf einer Website. Dieser kann benutzt werden, um mittels JavaScript eine Zeichnung in diesem Bereich erscheinen zu lassen.

Hier ein Beispiel einer mittels **`svg`**-Element erzeugten Grafik, und der entsprechende Quellcode:

Abbildung 5.1 - Eine mittels SVG-Element erzeugte Grafik und der entsprechende Quellcode

Einbinden von Audio-Dateien

Mittels **`audio`**-Element können nun auch Audio-Dateien in eine Website eingebunden werden. Dabei werdend die Formate MP3, WAV und OGG unterstützt.(vgl. [W3S19, Audio])

Das **`audio`**-Element selbst dient nur als Container für **`source`**-Elemente, welche die Referenzen auf die Audio-Dateien enthalten. Diese wird durch das **`src`**-Attribut, in Form einer URL oder eines relativen Pfades gesetzt. Dazu muss noch durch das **`type`**-Attribute, der jeweilige Typ der Audiodatei gekennzeichnet werden. Dabei steht **„`audio/mpeg`"** für mp3-Dateien, **„`audio/ogg`"** für OGG-Dateien und **„`audio/wav`"** für WAV-Dateien.

Durch das deklarieren des **`controls`**-Attributes in dem Tag des **`audio`**-Elements, wird ein Start-, ein Stopp- und ein Lautstärkeregelung-Button hinzugefügt. Diesem Attribut muss kein Wert zugewiesen werden.

Abbildung 5.2 - Der Code und die Darstellung eines eingebundenen Audio-Elements

Einbinden von Videos

Durch ein **`video`**-Element können nun auch Video-Dateien in eine Website eingebunden wurden. Dabei werden die Formate MP4, WebM und Ogg unterstützt. (vgl. [W3S19, Video])

Genau wie bei dem audio-Element wird durch das video-Element ein Container für **`source`**-Elemente definiert. Auch die Referenzierung der Datei erfolgt nach demselben Prinzip, jedoch muss bei dem **`type`**-Attribut auch ein entsprechendes Videoformat angegeben werden. (**`video/mp4`**, **`video/webm`**, **`video/ogg`**).

Wie auch bei dem audio-Element stellt das Attribut **`controls`** einen Videoplayer zur Verfügung. Dessen Größe kann über die Attribute **`width`** und **`height`** angepasst werden.

Hier ein Beispiel dazu:

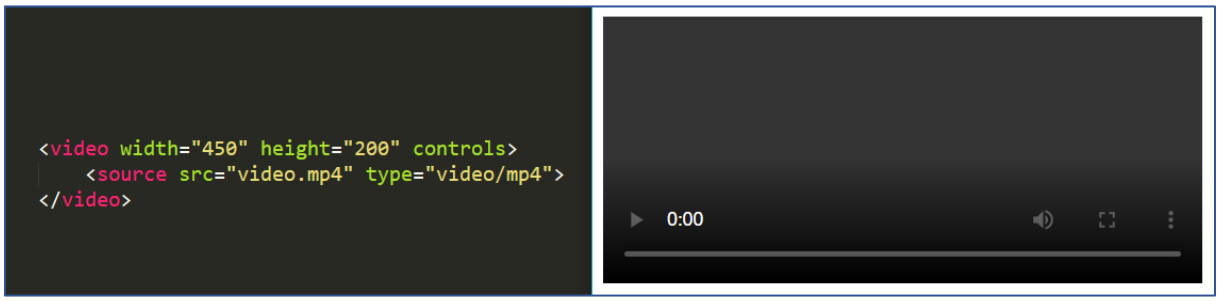

Abbildung 8 - Ein durch ein video-Element erzeugter Videoplayer, für die Datei „video.mp4“

Zusammenfassung

In diesem Kapitel wurden ihnen zu Beginn einige neue HTML5-Elemente vorgestellt, welche es ermöglichen bestimmte Elemente eines Bereiches zu gruppieren und dadurch die Struktur einer Website noch präziser festzulegen.

Anschließend wurden ihnen noch die Multimedia-Elemente vorgestellt, die es ermöglichen Video- und Audio-Dateien in eine Website einzubinden und SVG-Dateien zu erzeugen.

Aufgaben zur Selbstüberprüfung

Aufgabe 5.1:

Wozu wird das header-Element, und wozu das footer-Element verwendet?

Aufgabe 5.2:

Definieren sie ein Element das die Beispieldatei „video.mp4“ einbindet, und einen Video-Player der Größe 200x200 bereitstellt.

A. Lösungen der Aufgaben zur Selbstüberprüfung

Aufgabe 1.1:

Die Sprache HTML wird benutzt, um die grundlegende semantische Struktur einer Website festzulegen.

Aufgabe 1.2:

Da HTML der Standard der Webentwicklung geworden ist, basiert heutzutage nahezu jede Website auf einem HTML-Quellcode. Des Weiteren ist heutzutage nahezu jeder Browser darauf ausgelegt HTML-Quellcode zu Interpretieren.

Aufgabe 1.3:

Es ist nicht möglich, weitgehende Einstellungen an der visuellen Darstellung der Website festzulegen, oder dynamische Funktionen einzubauen.

Aufgabe 1.4:

HTML-Elemente sind Bestandteil eines HTML-Quellcodes, welche einen bestimmten Teil einer Website repräsentiert. Sie sind nötig um den HTML-Quellcode sowie die daraus resultierende Website strukturieren zu können, und dem Browser Interpretationsanweisungen zu geben.

```
<Tag-Name> Inhalt des Elements </Tag-Name>
```

Ein HTML-Element ist durch ein Start- und ein End-Tag ausgezeichnet, und zwischen diesen Tags befindet sich der Inhalt des Elements.

Aufgabe 2.1:

Die Dokumenttypdeklaration - **`<!DOCTYPE HTML>`**

Der HTML-Container - **`<html>`**

Die Unterteilung des HTML-Containers in **`<head>`** (Metainformationen) und **`<body>`** (Inhalt).

Der Titel der Website im head - **`<title>`**

Aufgabe 2.2:

Durch die Angabe des Zeichensatzes: **`<meta charset="UTF-8">`**

Aufgabe 2.3:

```
<!DOCTYPE html>
<html>
 <head>
  <title> </title>
 </head>
 <body> </body>
</html>
```

Aufgabe 2.4:

Die Elemente **<h1>-<h6>** und **<p>**, werden benötigt, um die einzelnen Überschriften und Paragrafen auszuzeichnen.

Zur Hervorhebung von Texten dienen die Elemente ** ** und **<small>.**

Aufgabe 2.5:

Listen können durch **<ul>** , **<ol>** oder **<dl>** definiert werden. Der Unterschied zwischen **ul** und **ol** ist, dass **ol** eine Aufzählung enthält und **ul** nicht. Der Aufbau dieser zwei Listen ist aber gleich.

Die **dl** hingegen unterscheidet sich im Aufbau zu den anderen Beiden und enthält zu jedem Listen-Term auch noch eine Term-Beschreibung.

Aufgabe 2.6:

```
<img src="bild.jpg" alt="Alternativtext">
```

Dem Attribut src kann ein relativer Pfad zu der Datei, oder eine URL zugewiesen werden.

Aufgabe 2.7:

Referenzen werden durch das anchor-Element gesetzt:

```
<a href="https://www.eine-website.de/einDokument.html" … >
Ich bin ein Link</a>
```

Für einen Sprung muss ein Element der Website durch eine **id**/**div** gekennzeichnet werden, und anschließend durch ein **#** angeführt, in das **href**-Attribut eingetragen werden.

Aufgabe 2.8:

<table> : Leitete eine Tabelle ein, Container für weitere Tabellen-Elemente

<tr> : Definiert eine neue Tabellenreihe.

<th> : Definiert eine neue Tabellenspalte, und deren Überschrift.

<td> : Definiert die eigentlichen Tabelleneinträge.

Aufgabe 3.1:

Formulare werden durch **<form>** definiert, und Eingabefelder durch **<input>**.

Beispiele: text, date, email, number, color, submit, reset etc.

Aufgabe 3.2:

Da die Datenübertragung in Form von „Schlüssel-Wert-Paaren“ stattfindet, muss jedem **input**-Element, durch das Attribut **name** auch ein bestimmter Schlüssel zugewiesen werden.

Des Weiteren muss dem form-Element mit den Attributen **action** und **method**, das Ziel der Datenübertragung, sowie deren Methode genannt werden.

```
<form action="formularVerarbeitung.php" method="post">
<input type="text" name="Name">
```

Aufgabe 4.1:

Ein DOM wird das aus den Elementen, den Attributen und den Beziehungen zwischen den einzelnen Elementen abgeleitet und aufgebaut. Es wird durch eine Baumstruktur repräsentiert und stellt die logische Gesamtstruktur eines HTML-Dokuments dar.

Aufgabe 4.2:

Das DOM definiert eine Programmierschnittstelle um Zugriffe und Manipulation, durch externe Skript- oder Programmiersprachen, zu ermöglichen. Dadurch kann der Inhalt einer Website dynamisch angepasst werden.

Aufgabe 4.3:

Die Elemente sollten durch bestimmte Klassen oder id’s ausgezeichnet werden, da Methoden bereitgestellt werden die gezielt darauf zugreifen können

Aufgabe 5.1:

Das header-Element wird üblicherweise verwendet, um Elemente zu definieren die konstant im oberen Bereich der Website platziert werden. Das footer-Element für Elemente im unteren Bereich.

Aufgabe 5.2:

```
<video width="200" height="200" controls>
        <source src="video.mp4" type="video/mp4">
</video>
```

B. Literaturverzeichnis

Kapitel 1:

[RL] Dave Raggett, Jenny Lam, Ian Alexander and Michael Kmiec: „Ragett on HTML4: Chapter 2: A history of HTML“; Addison Wesley Longman; 1998; ISBN: 0-201-17805-2; <https://www.w3.org/People/Raggett/book4/ch02.html> (12.11.2019).

[Be19] World Wide Web Consortium: Tim Berners Lee: Longer Biography; 2019; <https://www.w3.org/People/Berners-Lee/Longer.html> (12.11.2019).

[Se18] SELFHTML: Wiki: SGML; 2018; < https://wiki.selfhtml.org/wiki/SGML> (13.11.2019).

Kapitel 2:

[MK07] Chuck Musciano, Bill Kennedy: „HTML & XHTML: das umfassende Handbuch“; O'Reilly Germany; 2007; ISBN: 978-3897214941

[Sc08] Ralph G. Schulz: „HTML und CSS Praxisbuch: Einführung in strukturiertes Webdesign“; mitp Verlag; 2008; ISBN: 978-3826617751

[Pr09] Christoph Prevezanos : „Jetzt lerne ich HTML; Pearson Deutschland GmbH“; 2009; ISBN: 978-3827244383

[Se18] SELFHTML: Wiki: HTML/Tutorials/HTML5/Grundgerüst; 2018; <https://wiki.selfhtml.org/wiki/HTML/Tutorials/HTML5/Grundgerüst> (15.11.2019).

[W3S19] W3Schools: HTML Basic Examples; 2019; <https://www.w3schools.com/html/html_basic.asp> (15.11.2019).

[WH19] WHATWG: HTML Living Standard: The elements of HTML; 2019; <https://html.spec.whatwg.org/multipage/semantics.html#semantics> (15.11.2019)

[W3S19, Dt] W3Schools: HTML <!DOCTYPE> Declaration; 2019; <https://www.w3schools.com/tags/tag_doctype.asp> (15.11.2019).

[Se18] SELFHTML: Wiki: HTML/Tutorials/Einstieg/Kapitel2; 2019; <https://wiki.selfhtml.org/wiki/HTML/Tutorials/Einstieg/Kapitel2> (17.11.2019).

[W3S19] W3Schools: HTML Lists; 2019; <https://www.w3schools.com/html/html_lists.asp> (17.11.2019).

[W3S19] W3Schools: HTML <img> Tag; 2019; <https://www.w3schools.com/tags/tag_img.asp> (17.11.2019).

[W3S19] W3Schools: HTML Images; 2019; <https://www.w3schools.com/html/html_images.asp> (17.11.2019).

[W3S19] W3Schools: HTML File Paths; 2019; <https://www.w3schools.com/html/html_filepaths.asp> (17.11.2019).

[Se19] SELFHTML: Wiki: HTML/Multimedia und Grafiken/Grafiken; 2019; <https://wiki.selfhtml.org/wiki/HTML/Multimedia_und_Grafiken/Grafiken> (17.11.2019).

[W3S19] W3Schools: HTML Links; 2019; <https://www.w3schools.com/html/html_links.asp> (18.11.2019).

[W3D] W3docs: HTML: How to Create an Anchor Link to Jump to a Specific Part of a Page; 2019; <https://www.w3docs.com/snippets/html/how-to-create-an-anchor-link-to-jump-to-a-specific-part-of-a-page.html> (18.11.2019).

[WH19] WHATWG: HTML Living Standard: The a element; 2019; <https://html.spec.whatwg.org/multipage/text-level-semantics.html#the-a-element> (18.11.2019).

[W3S19] W3Schools: HTML Tag; 2019; <https://www.w3schools.com/tags/tag_div.asp> (18.11.2019).

[W3S19] W3Schools: HTML Tables; 2019; <https://www.w3schools.com/html/html_tables.asp> (19.11.2019).

[Se19] SELFHTML: Wiki: HTML/Tabellen/Aufbau einer Tabelle; 2019; <https://wiki.selfhtml.org/wiki/HTML/Tabellen/Aufbau_einer_Tabelle> (19.11.2019)

Quelle des eingebundenen Bildes in Abb. 2.5 und Abb. 2.6:

[Re19] Pixabay: Megan_Rexazin: <https://pixabay.com/de/vectors/speicher-online-e-commerce-4156934/> (17.11.2019)

Kapitel 3:

[W3S19] W3Schools: HTML Forms; 2019; <https://www.w3schools.com/html/html_forms.asp> (20.11.2019).

[W3S19] W3Schools: HTML Form Elements; 2019; <https://www.w3schools.com/html/html_form_elements.asp> (20.11.2019).

[W3S19] W3Schools: HTML Input Attributes; 2019; <https://www.w3schools.com/html/html_form_attributes.asp> (20.11.2019).

[WH19] WHATWG: HTML Living Standard: The input element; 2019; <https://html.spec.whatwg.org/multipage/input.html#the-input-element> (20.11.2019)

[Se19] SELFHTML: Wiki: HTML/Tutorials/Formulare/Was ist ein Webformular?: Formular-Versand; 2019 <https://wiki.selfhtml.org/wiki/HTML/Tutorials/Formulare/Was_ist_ein_Webformular%3F#Formular-Versand> (20.11.2019).

Kapitel 4:

[PH04] Philippe Le Hégaret, W3C : What is the Document Object Model?, <https://www.w3.org/TR/DOM-Level-3-Core/introduction.html> (23.11.2019).

[W3S19] W3Schools: What is HTML DOM?; 2019; <https://www.w3schools.com/whatis/whatis_htmldom.asp> (22.11.2019).

[Se19] SELFHTML: Wiki: DOM; 2019; <https://wiki.selfhtml.org/wiki/DOM> (22.11.2019).

[W3C04] Philippe Le Hégaret, W3C: Document Object Model (DOM) Bindings; 2019; <https://www.w3.org/DOM/Bindings> (22.11.2019)

Kapitel 5:

[W3S19] W3Schools: HTML5 Introduction; 2019; <https://www.w3schools.com/html/html5_intro.asp> (23.11.2019).

[Se19] SELFHTML: Wiki: HTML/Tutorials/HTML5/Seitenstrukturierung; 2019; <https://wiki.selfhtml.org/wiki/HTML/Tutorials/HTML5/Seitenstrukturierung> (23.11.2019).

[W3S19] W3Schools: HTML5 Semantic Elements; 2019; <https://www.w3schools.com/html/html5_semantic_elements.asp> (23.11.2019).

[W3S19, SVG] W3Schools: HTML5 SVG; 2019; <https://www.w3schools.com/html/html5_svg.asp> (24.11.2019).

[W3S19, Canvas] W3Schools: HTML5 Canvas; 2019; <https://www.w3schools.com/html/html5_canvas.asp> (24.11.2019).

[W3S19, Audio] W3Schools: HTML5 Audio; 2019; <https://www.w3schools.com/html/html5_audio.asp> (24.11.2019).

[W3S19, Video] W3Schools: HTML5 Video; 2019; <https://www.w3schools.com/html/html5_video.asp> (24.11.2019).

C. Abbildungsverzeichnis

Abb. 2.1: Die HTML Grundstruktur **7**

Abb. 2.2: Textstrukturierung in HTML 8

Abb. 2.3: Eine ungeordnete Liste mit style-Attribut 9

Abb. 2.4: Eine geordnete Liste mit type- und start-Attribut 9

Abb. 2.5: Der Quellcode einer Definitionsliste und ihre Darstellung 10

Abb. 2.6: Einbindung eines Bildes mit einem Beschreibungstext und Maßangaben 11

Abb. 2.7: Referenzierung des Dokuments „main.html“, durch ein anchor um das Bild 12

Abb. 2.8: Die Darstellung einer HTML-Tabelle im Browser 14

Abb. 3.1: Die verschiedenen Input-Typen von Formularen 17

Abb. 3.2: Ein Kontaktformular mit Konfiguration zur Datenübertragung 18

Abb. 3.3: Ein Registrierungs-Formular dessen Quellcode 19

Abb. 4.1: Darstellung eines HTML-Dokuments in Baumstruktur 21

Abb. 5.1: Eine mittels SVG-Element erzeugte Grafik und der entsprechende Quellcode 25

Abb. 5.2: Der Code und die Darstellung eines eingebundenen Audio-Elements 25

Abb. 5.3: Ein durch ein video-Element erzeugter Videoplayer, für die Datei „video.mp4“ 26

D. Codeverzeichnis

Code 7.1 - Der Aufbau eines HTML-Elements 3
Code 8.1 – Dokumenttypdeklaration in HTML5 5
Code 2.2 - Dokumenttypdeklaration in HTML4.01 5
Code 2.9 - Das html-Element mit einem lang-Attribut 6
Code 2.10 – Ein title-Element innerhalb des head-Elements 6
Code 2 11 - Angabe des Zeichensatzes mit dem charset-Attribut innerhalb eines meta-Elements 6
Code 2.12 - Angabe des Autors und von Keywords mittels content- und name-Attribut. 7
Code 2.7 – Die möglichen Pfadangaben durch ein src-Attribut des img-Elements. 10
Code 2.8 – Die möglichen Pfadangaben beim Setzen einer Referenz durch das href-Attribut 11
Code 2.9 – Die Zuweisung einer id für ein h1-Element 13
Code 2.10 - Die Zuweisung einer id für ein div-Element 13
Code 2.11 - Das Festlegen eines Sprungs, durch eintragen einer id in einem href-Attribut 13
Code 2.12 – Die Struktur einer Tabelle in einem HTML-Quellcode 13
Code 3.1 - Die Deklaration eines label-Elements, welches mit einem input-Feld zusammenhängt 18
Code 3.2 - Das Festlegen eines default-Wertes für ein input-Element 18
Code 3.3 – Die Definition eines Buttons mit benutzerdefinierter JavaScript-Funktion 19