

Hinweis:

Diese Druckversion der Lerneinheit stellt aufgrund der Beschaffenheit des Mediums eine im Funktionsumfang stark eingeschränkte Variante des Lernmaterials dar. Um alle Funktionen, insbesondere Animationen und Interaktionen, nutzen zu können, benötigen Sie die On- oder Offlineversion. Die Inhalte sind urheberrechtlich geschützt.
©2018 Beuth Hochschule für Technik Berlin

HTM - Gestaltung von Ansichten mit HTML



Überblick und Lernziele

Die Lehrveranstaltung basiert auf der Annahme, dass Sie bereits grundlegende Kenntnisse in der Handhabung der Webtechnologien HTML, CSS und JavaScript erworben haben. Die vorliegende Lerneinheit wird daher darauf aufbauend einen Überblick über die Ausdrucksmittel vornehmen, die die aktuellen Versionen von HTML und CSS bereitstellen und insbesondere deren Anwendungsmöglichkeiten für die Entwicklung von Webanwendungen aufzeigen, die auf mobilen Geräten zum Einsatz kommen.

Den Schwerpunkt unserer Ausführungen bildet die Absicht, Ihnen die ggf. noch aufzubauenden Kompetenzen zu vermitteln, die Sie für die Bearbeitung des Übungsprogramms benötigen. Dort werden Sie die Ansichten, die in der von uns verwendeten Designvorlage für die mobile Anwendung „Müllers Welt“ konzipiert werden, partiell umsetzen.

Darauf aufbauend werden Sie in der nachfolgenden Lerneinheit „CSS - Graphische Oberflächengestaltung mit CSS“ die Ausdrucksmittel von JavaScript anwenden, um die zunächst nur mittels HTML und CSS umgesetzten statischen Ansichten der Anwendung in verschiedener Weise zu dynamisieren. Letzteres schließt den dynamischen Aufbau der Ansichten und den durch Nutzerinteraktion veranlassten Übergang zwischen Ansichten ohne Neuladen des verwendeten HTML-Dokuments ein.



Lernziele

Nachdem Sie die Lerneinheit durchgearbeitet haben, sollten Sie in der Lage sein:

- Die wesentlichen Punkte in der Entwicklung von HTML zu nennen.
- Die Grundzüge von HTML zu erklären und deren Ausdrucksmittel einzusetzen.
- Auszeichnungselemente zur semantischen Kennzeichnung von Inhalten einzusetzen und erklären zu können.
- Die Ausdrucksmittel von HTML in der Umsetzung einer Gestaltungsvorlage für mobile Geräte anzuwenden



Gliederung

Nach einer kurzen Darlegung der theoretischen Aspekte, die die Sprachen HTML und CSS als Programmiersprachen im weiteren Sinne jeweils kennzeichnen, werden wir die für die Zwecke unserer Lehrveranstaltung relevanten Ausdrucksmittel vorstellen. Für HTML beinhaltet dies auch aktuelle Ausdrucksmittel, die einen Bezug zur Entwicklung von Anwendungen auf mobilen Geräten aufweisen, die im Verlauf der Veranstaltung aber nicht näher betrachtet werden.

Nur am Rande eingehen werden wir - wo erforderlich - auf die Ausdrucksmittel des HTTP Protokolls, die die Grundlage für die durch HTML realisierbare Verknüpfung von Inhalten verschiedener Typen und aus verschiedenen Quellen darstellen. Diesbezüglich nehmen wir ebenfalls an, dass Sie bereits über einschlägiges Vorwissen aus anderen Veranstaltungen verfügen. Falls erforderlich, finden Sie in unserer Darstellung von Client-Server Architekturen in der Lerneinheit „COH - CRUD Operationen via HTTP mit NodeJS und MongoDB“ mit Blick auf die Veranlassung lesender und schreibender Datenzugriffsoperationen via HTTP in Kapitel 4 weitere Ausführungen zu den Ausdrucksmittel des Protokolls.



Zeitbedarf

Zeitbedarf und Umfang

Für die Bearbeitung der Lerneinheit benötigen Sie etwa 3 Stunden. Für die Bearbeitung der Übungen zur Beispielanwendung etwa 2.5 Stunden und für die Wissensfragen zur Lerneinheit ca. 1.5 Stunden.

1 HTML

Im folgenden gehen wir zunächst darauf ein, dass wir mit HTML eine Auszeichnungssprache verwenden und damit eine Sprache, deren Verwendung und Ausführung sich deutlich von anderen hier betrachteten Sprachen wie CSS und JavaScript unterscheidet. Aus einem kurzen historischen Rückblick auf die Entstehung von HTML werden wir dann die Grundzüge der aktuellen auch als HTML5 bezeichneten Version der Sprache ableiten und einige von deren Ausdrucksmitteln exemplarisch vorstellen. Eine Betrachtung der Verwendung von Auszeichnungselementen zur semantischen Kennzeichnung der Struktur von Ansichten, die mittels HTML aufgebaut werden, wird dann zu CSS überleiten, das dann Thema der nächsten Lerneinheit sein wird.



2 HTML als Auszeichnungssprache

Mit Auszeichnungssprachen haben Sie vermutlich bereits in den ersten Jahren Ihrer Bildungslaufbahn Bekanntschaft gemacht, nämlich in Form von rot geschriebenen Korrekturanmerkungen in Klassenarbeiten. Diese Anmerkungen beziehen sich auf den von Ihnen verfassten Text und stellen die Sichtweise des Korrigierenden bezüglich der Inhalte dieses Textes dar. Zu diesem Zweck können z. B. verschiedene Formen von Unterstreichungen (gerade Linien, Wellenlinien, etc.) Durchstreichungen, Einfügungskennzeichnungen etc. verwendet werden, die ggf. ergänzt werden um Symbole am Seitenrand oder natürlich sprachliche Anmerkungen.

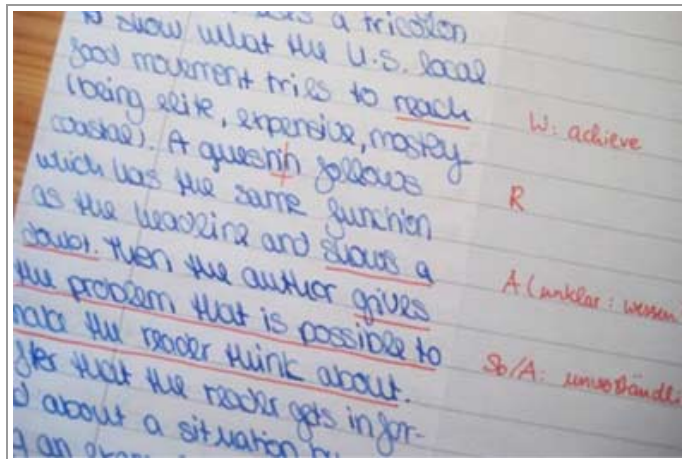


Abb.: Korrekturanmerkungen in Klassenarbeiten

Reservierte Farben

Vielleicht haben Sie sich damals auch gefragt, weshalb Sie zwar möglicherweise Schreibgeräte für verschiedenste Schriftfarben verwenden durften, die Rotschrift jedoch Ihren Lehrern vorbehalten war. Stellen Sie sich aber einmal vor, dass die Korrekturanmerkungen und Ihr eigener Text in der gleichen Farbe vorliegen. Könnten daraus nicht Missverständnisse bezüglich einzelner textueller Inhalte entstehen, die es im nachhinein erschweren, Ihren ursprünglichen Text und die Korrekturanmerkungen auseinander zu halten? So dient denn die Einführung von Rot als reservierter Farbe für Korrekturhinweise denn genau dazu, eine solche Problematik gar nicht erst aufkommen zu lassen, da sie gewährleistet, dass die ursprünglichen Inhalte eines Textes und die darauf bezogenen Korrekturhinhalte schon alleine aufgrund der Textfarbe unterschieden werden können und dafür nicht etwa der Einsatz eines Graphologen erforderlich ist. Willkürlich ist diesbezüglich nur die Entscheidung, Rot als eine solche reservierte Farbe zu verwenden. die Einführung einer reservierten Farbe – egal welcher – stellt jedoch eine Notwendigkeit im Hinblick auf den verfolgten Zweck dar.

Metainformation, Metadaten

Verallgemeinert lassen sich Auszeichnungssprachen auffassen als Sprachen, die das Hinzufügen von zusätzlichen Inhalten zu bestehenden Inhalten bei Gewährleistung der Unterscheidbarkeit beider Typen von Inhalten ermöglichen. Für solche zusätzlichen Inhalte wird stellenweise auch der Begriff Metainformation oder Metadaten verwendet, der aber nichts weiter zum Ausdruck bringt als die Tatsache, dass die hinzugefügten Inhalte sich auf die bestehenden Inhalte beziehen, bzw. dass es sich bei Ihnen um Inhalte über Inhalte handelt. Zweifelsohne trifft dies auf das betrachtete Beispiel von Korrekturanmerkungen im Schul- oder Hochschulwesen zu. Als anderes Beispiel aus der Berufswelt seien hier Textauszeichnungen im traditionellen Druckgewerbe genannt, mit denen in vergleichbarer Form Angaben zur druckerischen Wiedergabe eines Textes gemacht werden können und z. B. Merkmale wie Schrifttyp, Schriftgröße und Schriftschnitt auch einzelner Bestandteile des Textes festgelegt werden können – ein Beispiel hierfür sehen Sie in der folgenden Abbildung.

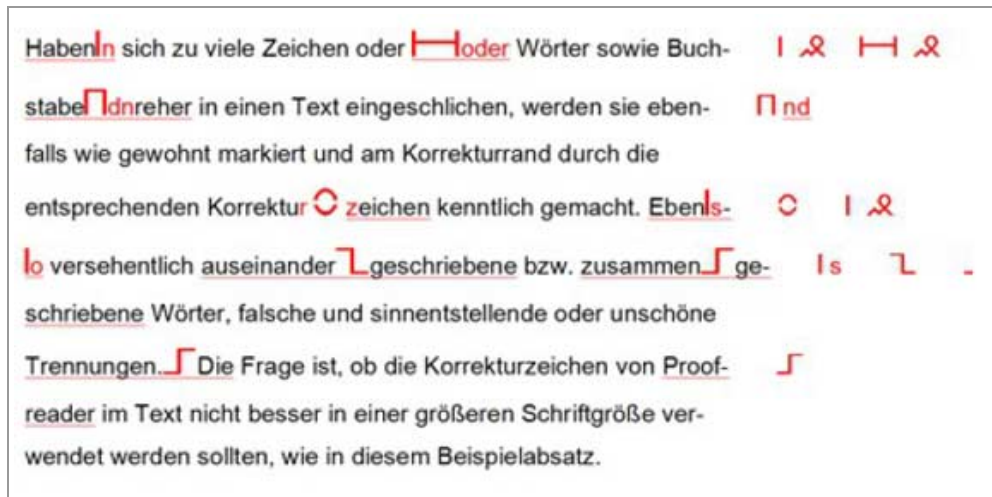


Abb.: Beispiel für „manuell lesbare“ Auszeichnungssprache

Ausdrucksmittel von Auszeichnungssprachen

Was aber sind die Voraussetzungen die für das Funktionieren einer Auszeichnungssprache gegeben sein müssen? Diese lassen sich bereits aus unserem eingangs genannten Beispiel schulischer Korrekturanmerkungen ableiten. Hier sehen wir, dass durch die Ausdrucksmittel einer Auszeichnungssprache insbesondere die folgenden Anforderungen berücksichtigt werden müssen:

- Die Ausdrucksmittel müssen die Unterscheidbarkeit von ursprünglichen Inhalten und Metainformation bzw. von ausgezeichneten Inhalten und auszeichnenden Inhalten ermöglichen.
- Die Ausdrucksmittel müssen die Zuordnung von auszeichnenden Inhalten zu ausgezeichneten Inhalten ermöglichen.
- Für auszeichnende Inhalte muss ein begrenztes Repertoire an Symbolen verwendet werden, mittels derer die Art der Auszeichnung ausgedrückt wird.

Die zuletzt genannte Anforderung findet sich im Bereich von Korrekturanmerkungen z. B. in Form der Vereinbarung verschiedener Symbole für verschiedene Fehlerarten, z. B. Ausdrucksfehler vs. Grammatikfehler vs. Rechtschreibfehler. Sie gewährleistet, dass eine Auszeichnungssprache für diejenigen verständlich ist, an die sich die Äußerungen in der betreffenden Sprache richten, d. h. in unserem Beispiel für Schüler als Adressaten einer korrigierten Klassenarbeit (und ggf. für deren Eltern). Vom Umfang der eingeführten Symbole hängt dann die Ausdrucksstärke einer Auszeichnungssprache ab, d. h. was alles unter Verwendung der betreffenden Sprache ausgesagt werden kann.

2.1 Maschinenlesbare Auszeichnungssprachen

Alle der oben genannten Anforderungen sind auch unabdinglich, wenn eine Auszeichnungssprache verwendet werden soll, um Aussagen zu treffen, die sich an eine Softwareanwendung als Adressaten wenden. In der Auszeichnungssprache LaTeX könnte eine Auszeichnung wie im folgenden Beispiel formuliert werden.



Beispiel

Ausdrucksmittel

Die Ausdrucksmittel müssen die `\emph{Unterscheidbarkeit}` von ursprünglichen Inhalten und `\scare{Metainformation}` bzw. von `\emph{ausgezeichneten}` Inhalten und `\emph{auszeichnenden}` Inhalten ermöglichen.

Mittels des Symbols „\“ wird zum Ausdruck gebracht, dass es sich bei den darauf folgenden Symbolen „`emph`“ und „`scare`“ um Symbole der Auszeichnungssprache, d. h. um auszeichnende Inhalte, handelt. Die Klammern „{...}“ zeigen an, auf welche ausgezeichneten Inhalte sich diese Inhalte beziehen. Die Symbole „`emph`“ und „`scare`“ sind sowohl für menschliche Leser als Adressaten des mit Auszeichnungen versehenen Textes verständlich, sofern sie mit LaTeX vertraut sind, als auch für die Softwareanwendung, der dieser Text zur automatischen Bearbeitung übergeben wird, um daraus beispielsweise ein PDF-Dokument zu erzeugen.

Symbol "`scare`"


Einschränkend sollte hier erwähnt werden, dass es sich bei "`scare`" um ein durch den Autor des Skripts definiertes Makro handelt, das an anderer Stelle definiert wird und das die Hinzufügung von einfachen Anführungszeichen um den ausgezeichneten Text herum zur Folge hat.

Verschiedene maschinenlesbare Auszeichnungssprachen unterscheiden sich nun nicht nur hinsichtlich des Umfangs an sprachspezifischen Symbolen, d. h. hinsichtlich ihrer Ausdrucksstärke, voneinander, sondern auch dadurch, wie diese Symbole zum Zweck der Auszeichnung angewendet werden. So stellen die nachfolgenden Beispiele dar, wie der Ausdruck **Fettschrift** in verschiedenen Auszeichnungssprachen – LaTeX, [MediaWiki](#) und HTML – realisiert werden kann. Hervorgehoben sind hier jeweils die Mittel, durch die die Unterscheidbarkeit von auszeichnenden Inhalten und ausgezeichneten Inhalten gewährleistet wird:

- `\textbf{Fettschrift}` (LaTeX)
- `'''Fettschrift'''` (MediaWiki)
- `Fettschrift` (HTML)

Soll andererseits eines der „Steuerzeichen“, die zum Zweck der Unterscheidbarkeit eingeführt werden, in den ausgezeichneten Inhalten selbst verwendet werden, dann muss diese Verwendung ihrerseits durch eine spezifische Markierung oder mittels spezifischer Zeichenfolgen gekennzeichnet werden. Beispielsweise dient in LaTeX die Zeichenfolge `\$` zur Darstellung des Steuerzeichens `$` und in HTML das Symbol `>` zur Darstellung des Steuerzeichens `>`.

2.2 XML

Im Hinblick auf die Findung geeigneter Ausdrucksmittel für maschinenlesbare textuelle Auszeichnungssprachen stellt die  Extensible Markup Language einen wichtigen Meilenstein dar, da XML als eine verallgemeinerte Lösung für die bereits genannten allgemeinen Anforderungen an Auszeichnungssprachen angesehen werden kann. Anders als es der Name suggeriert, ist XML selbst keine *konkrete* Auszeichnungssprache. Vielmehr definiert XML generische Ausdrucksmittel für die Repräsentation von Auszeichnungsinhalten in Textdokumenten, die zum einen die Unterscheidbarkeit von auszeichnenden und auszeichneten Inhalten, zum anderen die Zuordnung der Inhalte zueinander ermöglichen.

Dies erfolgt durch Definition eines Ihnen vermutlich vertrauten Zeicheninventars – `< , > , / , = , \` etc. – das Sie verwenden können, um die Symbole einer Auszeichnungssprache als solche zu kennzeichnen. Mittels dieser Zeichen können insbesondere Auszeichnungselemente – „Tags“ – gebildet und ggf. mit zusätzlicher Information mittels Attributzuweisungen versehen werden.

XML

Wir verwenden XML syntaktisch als Eigennamen für die eXtensible Markup Language bzw. die XML und verwenden in Bezug auf diese das Neutrum als grammatikalisches Geschlecht.

Syntaktische Regeln der XML

Bezüglich der Anwendung von Auszeichnungselementen formuliert die XML-Spezifikation eine Menge von Regeln, die erfüllt sein müssen, damit ein Text mit Auszeichnungselementen als *wohlgeformtes Dokument* im Sinne von XML angesehen und behandelt werden kann. Auch diese dürften Ihnen bereits vertraut sein – eine Auswahl sei daher nachfolgend nur in Kürze genannt:

- Ein XML Dokument hat *genau ein Wurzelement*.
- Jedes andere Element hat *genau ein Mutter-Element* und kann *beliebig viele Geschwister- und Kind-Elemente* haben.
- Jedem öffnenden Tag muss ein schließender Tag entsprechen: ` ... `
- Tags ohne Inhalt müssen mit der Zeichenfolge `
` geschlossen werden: `
`
- Attributwerte werden durch `"..."` umschlossen: `<input type="submit"/>`

Mittels dieser Regeln konzipiert XML einen Text mit Auszeichnungsinhalten also als eine *Baumstruktur*. Diese kann ausgehend von einem Wurzelement beliebig viele ineinander eingebettete Elemente enthalten, die ihrerseits jeweils den genannten Regeln entsprechen. Damit wird zugleich auch eine Grenze bezüglich der *geradlinigen* Anwendbarkeit von XML bezeichnet. Diese wird markiert durch Sachverhalte, die sich anhand der für einen Baum erforderlichen eindeutigen Mutter-Tochter Beziehungen ausdrücken lassen. Andere Sachverhalte sind zwar mittels XML ausdrückbar, jedoch nicht mehr auf Grundlage der Baumstruktur selbst, sondern durch Verwendung geeigneter anwendungsspezifischer Attribute, mittels derer z. B. Elemente des Baumes referenziert werden können.

Auszeichnungssprache

Inwiefern ist nun XML aber eine Auszeichnungssprache wie LaTeX, MediaWiki oder HTML? Diese erlauben doch jeweils die Auszeichnung von Inhalten zu einem bestimmten Zweck, nämlich zum Zweck der visuellen Darstellung dieser Inhalte. XML führt jedoch keine spezifischen Symbole ein, die zu einem solchen Zweck oder zu anderen Zwecken verwendet werden können. Man könnte auch sagen, dass XML selbst vollkommen zweckfrei bzw. offen für beliebige Anwendungszwecke ist. Diese Abstraktheit im Sinne einer Zweckungebundenheit, die XML kennzeichnet, führt nun immer wieder zu der Frage, inwiefern es sich bei XML überhaupt um eine Auszeichnungssprache handelt oder nicht.

XML als Auszeichnungssprache

Zur Illustration hierfür können Sie z. B. eine Google Suche bezüglich der Aussage "*Is XML a markup language*" oder auch "*XML is not a markup language*" durchführen.

Abstraktheit

Diesbezüglich hatten wir es uns ja bisher recht einfach gemacht, als wir XML als *abstrakte* Auszeichnungssprache eingeführt hatten. Um etwas präziser zu werden, könnte man die Aussage treffen, dass XML eine Auszeichnungssprache im Sinne einer *Sprache für die Notation von Auszeichnungsinhalten* ist und zu diesem Zweck ein Zeicheninventar und Regeln für dessen Anwendung einführt. Diese Ausdrucksmittel bilden aber erst in Verbindung mit einer Menge von Symbolen – z. B. *b, form, input, body, etc.* – eine konkrete Auszeichnungssprache, z. B. HTML.

Abstraktheit in Bezug auf XML bedeutet also, dass sich die Sprache immer nur in einer konkreten Auszeichnungssprache manifestiert, d. h. XML kann nicht für die Erstellung von Auszeichnungsinhalten verwendet werden, ohne dass zugleich eine konkrete Auszeichnungssprache verwendet wird – dies könnte man vergleichen mit der Nichtinstantiierbarkeit abstrakter Klassen in Java. XML ist also nicht nur, wie es der Name der Sprache besagt, *erweiterbar* (extensible), *sondern bedarf* im Hinblick auf seine Verwendung der *Erweiterung durch konkrete Auszeichnungssprachen*. Inwiefern HTML eine solche Erweiterung darstellt oder darstellte werden wir im folgenden Teilabschnitt darlegen.

Wohlgeformtheit und Validität

Im Gegensatz zur *Wohlgeformtheit* eines XML Dokuments, die alleine auf Basis der Regeln der XML entschieden werden kann, bedarf es für die Festlegung der korrekten Verwendung der Symbole einer konkreten auf XML basierenden Auszeichnungssprache zusätzlicher Regeln. Diese können in Form einer *Document Type Definition* oder eines *XML Schemas* maschinenlesbar festgehalten werden und für die maschinelle Überprüfung von XML Dokumenten verwendet werden. Entspricht ein wohlgeformtes Dokument diesen Regeln, dann wird es als *valide* in Bezug auf diese Regeln bezeichnet. Diese Erweiterung des generischen Konzept der *Wohlgeformtheit* eines Dokuments um den der *Gültigkeit* oder *Validität* eines Dokuments in Bezug auf eine konkrete Auszeichnungssprache und die Bereitstellung von Ausdrucksmitteln zur Formulierung solcher Gültigkeitsregeln erlaubt es, auf Basis von XML beliebige spezifische Auszeichnungssprachen zu definieren und Dokumente dieser Sprachen zur Laufzeit in einer Softwareanwendung, ggf. nach vorheriger maschineller Gültigkeitsprüfung, zu verwenden.

Softwareentwicklung mit XML

Die Generizität der XML erlaubt es, dass ihrerseits generische Softwarekomponenten sowohl für das Einlesen eines Textdokuments entsprechend den Wohlgeformtheitsregeln der XML, als auch für dessen Gültigkeitsprüfung eingesetzt werden können. Ein Anwendungsentwickler braucht also nur noch die anwendungsspezifische Behandlung der eingelesenen Inhalte zu implementieren und ist von der Aufgabe entlastet, eigene Komponenten für das Einlesen der Inhalte und ggf. deren Überprüfung entwickeln zu müssen. Nicht zuletzt braucht er sich auch nicht vor jeglicher Implementierung zu überlegen, in welcher Weise die grundlegenden Merkmale einer Auszeichnungssprache zur Unterscheidung von ausgezeichneten und auszeichnenden Inhalten realisiert werden müssen, da XML ihm auch diese Aufgabe abnimmt. Die Konzeption und Verwendung einer konkreten Auszeichnungssprache auf Basis von XML erfordert also nur zwei wesentliche Maßnahmen. Zum einen müssen Sie überlegen, in welcher Form die Ausdrucksmittel der XML von der zu konzipierenden Sprache verwendet werden sollen. Zum anderen müssen Sie Anwendungskomponenten implementieren, die die Inhalte eines mittels generischer Komponenten eingelesenen XML Dokuments im Sinne Ihrer Anwendung weiter verarbeiten.

Zur weiten Verbreitung, die die Verwendung von XML seit Ende der 90er Jahre erlebt hat, dürften die hier genannten Eigenschaften und die Verfügbarkeit entsprechender generischer Komponenten für zahlreiche Ausführungsumgebungen und Programmiersprachen in hohem Maße beigetragen haben. Ein Grund hierfür liegt jedoch auch in der Plattformunabhängigkeit von XML als einem textuellen Repräsentationsformat und als Alternative zu proprietären binären Formaten. Die Verbindung von XML mit der auf Basis nativer JVM Implementierungen gleichermaßen „plattformunabhängigen“ Sprache Java galt denn auch vor allem zu Beginn der Nuller Jahre geradezu als „Marriage made in Heaven“. So sah man darin eine wichtige Grundlage dafür, den durch das Internet grundsätzlich ermöglichten Datenaustausch zwischen verteilten Softwareanwendungen, die ggf. auf unterschiedlichen Plattformen betrieben werden, tatsächlich implementatorisch umzusetzen.

2.3 XML und HTML

Auch wenn HTML von seinem ersten Erscheinungsbild sehr stark an XML erinnert, ist der Zusammenhang der beiden nicht geradlinig. Dies liegt insbesondere daran, dass XML in der Mitte der 90er Jahre entwickelt und als Standard 1997 veröffentlicht wurde, während HTML zu diesem Zeitpunkt bereits eine nicht unbeträchtliche Verbreitung im Rahmen von Webanwendungen bzw. als Grundlage statischer Websites verzeichnen konnte. Die Syntax von HTML basierte dabei auf der Standard *Generalized Markup Language*, als deren restriktivere Weiterentwicklung XML angesehen werden kann. Für die ursprüngliche Version von HTML, die zu Beginn der 90er Jahre erarbeitet worden war, waren die Wohlgeformtheitskriterien von XML daher zu strikt. Offensichtlich werden die Abweichungen gegenüber XML u. a. anhand von Notationsalternativen wie den folgenden:

- nicht-schließende Tags: `
...
 ...`
- Notation boolescher Attribute: `<input type="text" disabled>`
- Attribute ohne Anführungszeichen: `<input type=text>`

1997 waren jedoch bereits zahlreiche Websites in Betrieb, die diese Notation für HTML verwendeten. Sie können sich bestimmt vorstellen, dass es in dieser Situation eine Herausforderung darstellt, einen Standard durchzusetzen, der zunächst einmal weniger Freiräume bei der Erstellung von HTML Dokumenten erlaubt.

Ein solcher Versuch wurde gegen Ende der 90er Jahre mit der Arbeit an XHTML dennoch unternommen. Dabei handelt es sich um eine den Anforderungen von XML konforme syntaktische Variante von HTML, für welche zudem u. a. eine www Document Type Definition sowie ein www XML Schema existiert, welches die Validitätskriterien für HTML Dokumente festlegt. Die Tatsache, dass beide Beschreibungen in verschiedenen Striktheitsgraden vorliegen, deutet jedoch bereits darauf hin, dass sich die Verfechter von XHTML der oben genannten Herausforderung durchaus bewusst waren.

Auch aufgrund der Erfordernis, dass Browser einen gewissen Spielraum bezüglich der Verarbeitung und Darstellung nicht wohlgeformten HTML Markups mitbringen sollen, hat sich XHTML nicht durchgesetzt. Die aktuelle Version von HTML, deren Ausdrucksmittel wir im nachfolgenden Abschnitt vorstellen werden, erlaubt denn auch wieder offiziell Notationsalternativen wie die oben genannten. Mit Blick auf die Implementierungsbeispiele sei jedoch angemerkt, dass der Autor dieses Skripts üblicherweise eine XML-konforme Notation den letzteren vorzieht. Falls Sie sicherstellen wollen, ob es sich bei einem Textdokument tatsächlich um ein HTML Dokument gemäß den aktuell gültigen Regeln handelt, können Sie dies mittels eines Validator-Dienstes des W3C überprüfen, der Ihnen u. a. das Upload von Dokumenten erlaubt: www <http://validator.w3.org>

Die Idee eines *vollständigen Dokuments*, die den XML Konzepten von *Wohlgeformtheit* und *Gültigkeit* zugrunde liegt, spielt jedoch auch in Bezug auf die Ausdrucksmittel von HTML eine gewichtige Rolle. So stellt HTML selbst nach wie vor keine Mittel zur Verfügung, um HTML Dokumente aus Teil-Dokumenten / *Dokument-Fragmenten* aufzubauen. Aus Softwareentwicklungssicht bietet HTML damit *keine* Möglichkeit der *Modularisierung* von Dokumenten zum Zweck der *Wiederverwendung*. Schon allein dadurch ist eine Indikation für den Einsatz server-seitiger oder – wie in der vorliegenden Lehrveranstaltung praktiziert – client-seitiger Verfahren zur dynamischen Markup-Generierung gegeben.

Wohlgeformtheit und Gültigkeit der tatsächlich zur Laufzeit verwendeten HTML Dokumente lassen sich damit jedoch nicht mehr zur Entwicklungszeit verifizieren, und hinsichtlich client-seitiger Verfahren bestehen verschiedene Alternativen, die sich in ihrer Handhabbarkeit, aber auch Fehleranfälligkeit erheblich voneinander unterscheiden – wir verweisen dafür auf die Ausführungen in der Lerneinheit „JSL - Interaktionssteuerung mit Javascript“. Zur Erleichterung von wohlgeformtem dynamischen Markup sieht HTML denn auch mit [www](#) Templates ein noch in Ausarbeitung befindliches Ausdrucksmittel vor, bei dem dynamisch mit Inhalten zu befüllende Markupfragmente innerhalb eines HTML Dokument notiert werden können und damit auch für eine Syntaxprüfung zur Entwicklungszeit zugänglich sind.


Bis hierher hat diese Lerneinheit Ihnen einige allgemeine Bemerkungen zur Syntax von Auszeichnungssprachen, zum Stellenwert, den XML diesbezüglich einnimmt, sowie zum Verhältnis von HTML und XML präsentiert, anhand derer Sie Ihre bereits vorliegenden Kenntnisse bezüglich HTML evtl. rekapitulieren konnten. Eine Erläuterung dessen, was HTML ist – abgesehen davon, dass es sich um eine Auszeichnungssprache handelt – und zu welchem Zweck es verwendet wird, haben wir Ihnen auch unter Annahme vorhandener Kenntnisse diesbezüglich bisher jedoch vorenthalten. Der Vermittlung dieses Wissens in Form eines Überblicks dient die nachfolgende Darstellung der Ausdrucksmittel von HTML, die auch die aktuellen als „HTML5“ bekannten Mittel vorstellen wird.

3 Ausdrucksmittel von HTML und HTML5

Im Gegensatz zu XML haben wir es bei HTML mit einer *konkreten Auszeichnungssprache* zu tun. Solche Sprachen – erwähnt hatten wir oben z. B. auch LaTeX und die von Media Wiki verwendete Auszeichnungssprache – lassen sich aber nicht nur hinsichtlich ihrer syntaktischen Merkmale, sondern auch in Bezug auf ihrer Verwendungszweck unterscheiden. So handelt es sich bei Auszeichnungssprachen im Gegensatz zu nahezu uneingeschränkt verwendbaren Sprachen wie C oder Java denn zumeist um *domänenspezifische Sprachen* (DSLs), die für einen bestimmten Anwendungsbereich, Domäne genannt, konzipiert werden. LaTeX beispielsweise ist eine Auszeichnungssprache, die Autoren die semi-professionelle Formatierung von Texten bei gleichzeitiger Fokussierung auf die zu erstellenden Inhalte ermöglicht.

Was ist nun aber die Domäne von HTML?

HTML als DSL

Auskunft bezüglich dieser Frage erteilt uns der Name der Sprache selbst. So besagt *Hypertext Markup Language*, dass die ursprüngliche Domäne von HTML *Hypertext* ist, d. h. Text, dessen Eigenschaften über übliche Texteneigenschaften hinausgehen bzw. der Texte übergreift, indem er sie miteinander verknüpft und zusammenführt. Die Idee von Hypertext steht dabei im Kontrast zu einem Verständnis von Text als eine linear zu konsumierenden Folge von Aussagen, die für alle Konsumenten identisch ist – idealtypisch repräsentiert z. B. durch narrative Texte wie Erzählungen und Romane. Stattdessen nimmt Hypertext eine Verknüpfungen zwischen solchen linearen Texten vor und ermöglicht es den Konsumenten, nach Belieben von einem Text zum nächsten zu springen. Jeder Konsument konsumiert also ggf. seine individuelle Folge von Aussagen. Beispiele für das Auftreten von Hypertext in der dem Zeitalter elektronischer Medien vorangegangenen *Gutenberg-Galaxis*  [McL01] sind *Fußnoten* und *Querverweise* in wissenschaftlichen Publikationen oder Nachschlagewerken.

Im folgenden werden wir einen kurzen Überblick über die Entstehungsgeschichte und den Umfang der aktuell verfügbaren Ausdrucksmittel von HTML geben. Im Anschluss daran werden wir gezielt auf einen für die Eigenschaft von HTML als Auszeichnungssprache relevanten Aspekt eingehen und aufzeigen, welche Ausdrucksmittel aktuell für die Umsetzung *semantischen Markups* verfügbar sind. Dies wird im Rahmen einer Betrachtung zur Abgrenzung von „Struktur“ und „Gestaltung“ zum zweiten Schwerpunkt dieser Lerneinheit, der Verwendung von CSS, führen.

4 Historie von HTML

Ausdrucksmittel

Im Hinblick auf das Konzept von Hypertext stellte HTML zunächst insbesondere die folgenden Ausdrucksmittel bereit – wir beziehen uns hier auf die auf der CERN Website verfügbare [WWW Rekonstruktion der ersten Website](#), die u. a. eine [WWW Auflistung der damals existierenden HTML Elemente](#) umfasst:

- Markierung von Textfragmenten als mögliches Ziel von Verknüpfungen mittels ``
- Verknüpfung zu einem Text und Textfragmenten aus einem oder verschiedenen HTML-Dokumenten mittels ``

In Erwägung gezogen wurde damals bereits, die Art der Verknüpfung, d. h. die semantische Beziehung zwischen Ursprungs- und Zieldokument der Verknüpfung, kenntlich zu machen, z. B. auszudrücken, ob eine Verknüpfung eine „Vertiefung“ vornimmt, einen „Kontrast“ zum Ausdruck bringt, oder eine „Analogie“ aufzeigt. Die semantische Auszeichnung der Elemente eines Textes stellt denn auch einen weiteren Schwerpunkt dar, der aus dem initialen Funktionsumfang von HTML abzulesen ist, so konnten z. B. die folgenden Aspekte einer Textstruktur mittels geeigneter Elemente kenntlich gemacht werden:

- Dokumenttitel: `<title>`
- Überschriften: `<h1>` - `<h6>`
- Absätze: `<p>`
- Aufzählungen und Listen: `` + ``

In seiner Urfassung erscheint HTML damit als eine Markupsprache für Hypertext, die mittels geeigneter Elemente die Eigenschaften von Textdokumenten und die Verknüpfung von Texten zu beschreiben ermöglicht. Vorausgesetzt dabei war zugleich die Verfügbarkeit eines als *Browser* bezeichneten Lesegeräts für diese Texte. Dieser erlaubt zum einen die Umsetzung der strukturellen Merkmale eines Textes in visuelle Merkmale, die z. B. durch Schriftgröße und Schriftschnitt sowie durch Einrückungen und Zwischenräumen die Struktur der Texte für deren Leser kenntlich machen und damit die Lektüre erleichtern. Zum anderen kommt dem Browser die Aufgabe zu, die Verknüpfung von Texten zur Laufzeit vorzunehmen.

Beispielsweise sollte bei Interaktion des Nutzers mit der Darstellung des folgenden Elements das im `href` Attribut referenzierte Dokument durch den Browser geladen und seinerseits zur Darstellung gebracht werden:



Quellcode

href Attribut

```
001 <a href="http://info.cern.ch/hypertext/WWW/TheProject.html">
002   Browse the first website
003 </a>
```

Verknüpfungen


Als technische Grundlage jeglicher Verknüpfungen bzw. der Verknüpfbarkeit zur Laufzeit, diente und dient in HTML das HTTP Protokoll, mittels dessen die Regeln des Zugriffs auf die durch URLs – *Uniform Resource Locators* – bezeichneten Dokumente und andere Inhalte beschrieben wird.

Diese ursprünglichen Ausdrucksmittel von HTML wurden im Verlauf der 90er Jahre ergänzt um weitere Elemente. Dazu gehören zum einen Elemente zur Auszeichnung der Struktur textueller Inhalte, die z. B. mittels `` als hervorgehoben markiert oder mittels `<table>` in Form einer Tabelle aufeinander bezogen werden können. Andererseits wurden weitere Verknüpfungsmöglichkeiten bezüglich nicht textueller Inhalte bereit gestellt, die z. B. mit `` die Einbindung von Bildern in Dokumente oder mittels `<applet>` auch die Integration client-seitig ausführbarer Java-Anwendungen ermöglichen sollten, später generalisiert durch den Tag `<object>`.

Durch Verwendung von `<script>` Elementen konnten außerdem als Alternative zu letzteren client-seitig ausführbare Skripte in JavaScript deklariert oder referenziert werden. Allerdings beinhalteten die Erweiterungen auch Elemente, die eine explizite auf die Darstellung eines Dokumenten bezogene Auszeichnung ermöglichten, z. B. die Schriftschnitt-bezogenen Elemente `<i>` und `` oder die zur Markierung von Umbrüchen und Linien verwendeten Elemente `
` und `<hr>`.

HTML `<hr>` Tag

Eine strukturelle Erläuterung zu `<hr>` findet sich auf den Seiten der w3schools.com.

 http://www.w3schools.com/tags/tag_hr.asp.

Aktueller Stand

Was den aktuellen Stand von HTML angeht, so existiert zur Kontrolle der Darstellung von HTML Elementen mit der Sprache CSS – *Cascading Style Sheets* – bereits seit 1996 eine deutlich ausdrucksstärkere Alternative zu den genannten darstellungsbezogenen Elementen von HTML. So wurde mit Verfügbarkeit von CSS sogar eine strikte Syntax für HTML formuliert, die die Verwendung dieser Elemente ausschließt. Die Bedeutung von client-seitig ausgeführtem JavaScript hat insbesondere seit Populärwerden von AJAX Implementierungsmustern seit ca. 2005 kontinuierlich zugenommen – entsprechend abgenommen hat der Gebrauch von Applets und (weniger stark) von *Flash* als client-seitig ausgeführten eingebetteten Anwendungen. Mit nativen mobilen Anwendungen ist Webanwendungen als bevorzugter Grundlage für User Interfaces, die den Zugriff auf Inhalte im WWW ermöglichen, allerdings seit 2007 auch ein ernsthafter Wettbewerber erwachsen.

Vor diesem Hintergrund erscheinen die im folgenden erläuterten Ausdrucksmittel, die für HTML aktuell existieren, in verschiedener Weise als konsequente Rückbesinnung auf die Ursprünge von HTML als rein deskriptiver Sprache, die der darstellungsunabhängigen Auszeichnung von Inhalten dient. Dies betrifft insbesondere weitere Ausdrucksmittel zur semantischen Auszeichnung der Inhalte von HTML Dokumenten, die insbesondere in Verbindung mit der aktuellen als CSS3 bekannten Version von CSS eine klare Separierung von *Struktur* und *Darstellung* ermöglicht. Darüber hinaus werden zur client-seitigen Steuerung des *Verhaltens* eines auf einem HTML Dokument basierenden User Interfaces mittels JavaScript weitere APIs bereit gestellt. Wie wir gleich sehen werden, lassen sich diese jedoch insbesondere auch in Bezug zur erwähnten Herausforderung von Web Technologien durch native mobile Anwendungen bringen.

5 HTML5

Unter dem Begriff „HTML5“ werden üblicherweise die aktuellen Ausdrucksmittel von HTML verstanden, die wir im folgenden in einem kurzen Überblick darlegen werden. Diese umfassen zum einen die Definition zusätzlicher Markup Elemente. Für diese werden zum anderen ergänzende JavaScript APIs definiert, ohne welche die Elemente teilweise auch nicht sinnvoll verwendet werden können, z. B. im Fall des `<canvas>` Elements. „HTML5“ beinhaltet aber auch JavaScript APIs, die nicht bezogen sind auf ein spezifisches Markup Element, sondern Funktionalität des Endgeräts verfügbar machen, auf dem eine Anwendung betrieben wird, z. B. die Möglichkeit, auf den aktuellen Standort eines Geräts zuzugreifen.

Nicht zuletzt aus dieser Situation ergibt sich aber eine grundlegende Begriffsunschärfe dahingehend, ob „HTML5“ die Gesamtheit aller *neuen* Ausdrucksmittel für Markup und JavaScript bezeichnet oder ob darunter ausschließlich die Markup-Erweiterungen verstanden werden sollen. Zur ansatzweisen Klärung dieser Frage wollen wir im folgenden einen kurzen Blick auf die „Historie von HTML5“ werfen.

Historie


Ausgangspunkt für HTML5 stellen die bereits 1999 bzw. 2002 verabschiedeten Standards HTML 4 und XHTML 1.0 als letztgültige Standards für HTML dar, die als „Recommendation“ durch das W3C verabschiedet wurden. Parallel zur Arbeit an XHTML 2.0 durch das W3C, wurde ein 2005 als Web Applications 1.0 veröffentlichter Alternativentwurf durch eine Gruppe von Browserherstellern erarbeitet, die unter dem nicht wirklich intuitiven Akronym „WHATWG“ (Web Hypertext Application Technology Working Group) bekannt ist. Dieser Gruppe gehören u. a. Apple, Mozilla, Opera und Google an.

Diese Konfliktsituation zwischen W3C und WHATWG führte schließlich zur Aufgabe der Erarbeitung von XHTML 2.0 durch das W3C im Jahr 2009, die realistischere wohl auch als Ende des Wunsches nach einer strikten XML Syntax für HTML angesehen werden kann. Seither wird die HTML5 Spezifikation gemeinsam durch W3C und WHATWG gemeinsam entwickelt, Treiber dieses Prozesses ist dabei die WHATWG. Für das Jahr 2014 war ursprünglich die Verabschiedung des Standardvorschlags als „Recommendation“ durch das W3C vorgesehen.

























Die Standardisierung von HTML5 erfolgt jedoch als ein „Work in Progress“ und wird begleitet durch die fortwährende ggf. partielle Umsetzung von im Standard vorgesehen oder darin einzubringenden Funktionen durch die Browserhersteller und die Verwendung dieser Funktionen durch die Anwendungsentwickler. Als Alternative zur Fixierung einer Versionsbezeichnung und deren Assoziation mit einem fest definierten Funktionsumfang verwendet die daher für WHATWG ihr tagesaktuelles (!) [www Spezifikationsdokument](#) die dieser Situation wohl angemessene Bezeichnung des *HTML Living Standard*. Das hält dagegen nach wie vor an der Bezeichnung HTML5 fest, wobei die betreffende [www Spezifikation](#) großenteils wortlautidentisch mit dem Dokument der WHATWG ist.

Der durch das W3C gewählte Untertitel „A vocabulary and associated APIs for HTML and XHTML“ unterstreicht dabei nicht nur das Festhalten des W3C am Ideal einer XML Konformität. Er macht zugleich deutlich, dass diese neue Version von HTML über die Spezifikation einer Markupsprache hinausgeht und insbesondere die eingangs erwähnten JavaScript APIs beinhaltet.

Seit Beginn der Arbeiten an HTML5 wurden erhebliche Änderungen hinsichtlich Funktionsumfang und Ausdrucksmitteln vorgenommen, die auch Entscheidungen bezüglich der Inklusion von Funktionen im Standard vs. der Auslagerung in eine separate Standardisierungsinitiative beinhalten. Es bestehen daher erhebliche Divergenzen zwischen dem aktuellen Stand der offiziellen Standardisierungsdiskussion und veröffentlichten Handbüchern, Web Tutorials und Entwicklerdiskussionen, wie auch in Bezug auf die Medienberichterstattung.

Aus Sicht der Anwendungsentwicklung spielt jedoch die Frage, in welchem Rahmen die Spezifikation und Standardisierung von Ausdrucksmitteln stattfinden, eine untergeordnete Rolle gegenüber der Verfügbarkeit dieser Ausdrucksmittel und ihrer Unterstützung durch möglichst viele Browserhersteller. Die von uns an dieser Stelle in der folgenden Tabelle angebotene Übersicht geht diesbezüglich auf eine umfangreiche Recherche im Jahr 2011 zurück, die wir hinsichtlich der Verknüpfung mit den offiziellen Spezifikationsdokumenten aktualisiert und um die jeweils von **GASSTON**  [Gas13] abgedeckten Aspekte ergänzt haben.

Gerade auch mit Blick auf die hohe Abdeckung durch letztere – aktuelle – Publikation erscheint die Übersicht nach wie vor als repräsentative Darstellung dessen geeignet, was in der Entwickler-Community unter HTML5 verstanden wird. Dies gilt auch für die Implementierungsbeispiele, die wir zur Illustration eines Teils der Funktionen bereitstellen – eine Sichtung verschiedener Tutorials im Herbst 2013 lässt uns auch dies als angemessen erscheinen.

Bezeichnung	Funktion	Ebene	Pilgrim	Koch	Öggl	Gasston	Spez.
 <u>Canvas API</u>	Skriptbasierte 2D Graphik	JavaScript	+	+	+	+	 *
 <u>SVG</u>	2D Vektorgraphik	Markup	-	-	+	+	 *
 <u>Lokaler Speicher</u>	Speicherung von nutzerspezifischen Daten auf dem Endgerät	JavaScript	+	+	+	+	 *
 <u>Offline Applications</u>	Lokale Speicherung von Markupseiten und anderen Ressourcen	JavaScript	+	+	+	(+)	 *
 <u>Geolocation API</u>	Nutzung der Positionsbestimmung des Endgeräts	JavaScript	+	+	+	+	 *
 <u>Web Sockets</u>	Push-Zugriffe von Servern auf Endgeräte	JavaScript	-	-	+	-	 *
 <u>Web Workers</u>	Langläufige Hintergrundoperationen	JavaScript	+	+	+	-	 *
 <u>Formulare</u>	Funktionserweiterungen für Webformulare, inkl. Formularvalidierung	Markup	(+)	+	+	+	 *
 <u>Drag & Drop</u>	Drag & Drop Unterstützung für GUIs	JavaScript	-	+	-	+	 *
 <u>Multimedia</u>	Nutzung von Video und anderen Medienressourcen ohne Browser-Plugins	JavaScript	+	+	+	+	 *
 <u>Textgliederung</u>	Semantische Kennzeichnung der strukturellen Konstituenten eines HTML body Elements (z. B. article, section, header, footer, nav, etc.)	Markup	-	+	+	+	 *
 <u>Microdata</u>	Semantische Annotation von Inhalten	Markup	+	+	+	+	 *

Die Tabelle zeigt eine Übersicht der Ausdrucksmittel, die auf Grundlage der genannten Publikationen mit HTML5 assoziiert werden können. Die Spalte Spezifikation enthält jeweils eine Verknüpfung zum W3C Spezifikationsdokument auf dem Stand von März 2014. Die unterstrichenen Funktionen verknüpfen zu den Implementierungsbeispielen unter der Annahme, dass diese durch einen unter 127.0.0.1:8020 laufenden Webserver, z. B. den in Aptana eingebauten Webserver, bereitgestellt werden.

Mehrwerte

Die Mehrwerte, die eine Standardisierung der Ausdrucksmittel für die beschriebenen Funktionen mit sich bringt, sind offensichtlich. Zum einen reduzieren sie die Notwendigkeit, proprietäre ggf. browserspezifische Skriptlösungen zu implementieren, z. B. im Fall der Verwendung von Drag&Drop. Zum anderen erlauben sie es, proprietäre Plugins durch browserunterstützte Funktionen abzulösen. Darüber hinaus sind z. B. für die Umsetzung von Formularen oder die Einbindung von Multimediaelementen Ausdrucksmittel auf Ebene des Markups verfügbar, die die Nutzung generischer durch den Browser bereitgestellter Bedienelemente ermöglichen und damit die Schwelle für die Nutzung der besagten Funktion in einer Anwendung herabsetzen.

Wie wir im weiteren Verlauf der Lehrveranstaltung sehen werden, werden die betreffenden Markup Elemente jedoch ergänzt um JavaScript APIs, die eine individuelle Nutzung der betreffenden Funktionen im Rahmen der Realisierung eines anwendungsspezifischen User Interfaces ermöglichen.

Kommerzielle Implikationen

Erwähnt werden sollen an dieser Stelle die Vorteile, die eine Standardisierung von Ausdrucksmitteln zur Anwendungsentwicklung mit sich bringt – auch wenn diese nicht spezifisch für die Entwicklung von Webanwendungen auf Basis der aktuellsten Standards sind. So wirken sich Standards auf mehrere Kennzahlen aus, die im Hinblick auf die Entwicklung von Softwareanwendungen relevant sind. Sie reduzieren nämlich nicht nur die Kosten für die Anwendungsentwicklung durch geringere Entwicklungs- und Testaufwände, sondern damit verbunden auch die Zeit, die für die Umsetzung einer Neuentwicklung oder Funktionserweiterung veranschlagt werden muss.

Für ein Unternehmen, das eine Softwarelösung einsetzen möchte – sei es auf Basis von Eigenentwicklung oder durch Auftragsvergabe – reduzieren sich damit sowohl die Amortisationsdauer (*Return on Investment*), nach der die Entwicklungsaufwände durch Kosteneinsparungen, Produktivitätszuwachs oder Umsatzwachstum aufgewogen werden, als auch die *Time to Market*, innerhalb derer die betreffende Lösung den Kunden, Partnern oder Mitarbeitern des betreffenden Unternehmens angeboten werden kann. Im speziellen Fall von HTML dürfte die Verwendung standardisierter Bedienelemente für die Nutzer einer Anwendung aber auch eine Reduktion der kognitiven Aufwände (*Cognitive Load*) mit sich bedeuten, die für die erforderliche Bedienung einer Anwendung aufzubringen sind, und dürfte damit einen Beitrag zur Barrierereduktion in Bezug auf die UI Nutzung leisten. Inwiefern letztere Annahme angesichts der Möglichkeit einer individuellen Anpassung der betreffenden UI Funktionen realistisch ist, sei jedoch dahingestellt.

Legacy Technologie

Angemerkt sei außerdem, dass ein „Green Field Szenario“, bei dem die Anwendungsentwicklung den aktuellen Standard und die diesen Standard unterstützenden Browser als Ausgangspunkt annehmen kann, in der Realität nur selten anzutreffen, da dafür die Anzahl und das Alter der auf Websites zugreifenden Endgeräte und die darauf laufenden ggf. historischen Browser zu unübersehbar sind. De facto Erweiterungen der standardisierten JavaScript APIs wie jQuery, die browserübergreifend nutzbare APIs bereit stellen, werden daher noch auf längere Sicht bei der Umsetzung einer Anwendung in Erwägung zu ziehen sein.

Mobile Webanwendungen

Zwar weisen auch mobile Endgeräte und Betriebssysteme sowie die darauf verwendeten Browser insbesondere bei Betrachtung von Android mittlerweile ein hohes Maß an Fragmentierung auf. Gerade für die Entwicklung mobiler Anwendungen erscheint die Verfügbarkeit der oben genannten Ausdrucksmittel jedoch von großem Interesse – mögen sie zur Erzielung einer möglichst großen Reichweite einer Ergänzung um unterstützende Bibliotheken bedürfen oder nicht. So lässt sich denn ein großer Teil der genannten Ausdrucksmittel in unmittelbaren Bezug zu Funktionen nativer mobiler Anwendungen setzen, für die bis vor kurzem noch gar keine vertretbare Lösung im Bereich von Webanwendungen zur Verfügung stand.

Die nachfolgende Übersicht unterstreicht sehr deutlich unsere oben getroffene Annahme, dass HTML5 auch als Antwort auf die Herausforderung von Webtechnologien durch native mobile Anwendungen angesehen werden kann:

Bezeichnung	?	Mehrwert
Canvas API/SVG	+	Animationen ohne proprietäre Plugins
Lokaler Speicher	+	Anwendungen werden auf dem Endgerät „installiert“ und sind ohne Netzverfügbarkeit nutzbar („App Paradigma“)
Offline Applications		
Geolocation API	+	Standortbasierte Dienste
Web Sockets	+	Push-Datenübermittlung an mobile Endgeräte
Web Workers	+	Asynchrone Operationen mit UI Feedback
Formulare	+	„Placeholder-Text.“ Standardisierte Formularvalidierung ohne proprietäre Skripts.
Drag & Drop	(–)	(HTML5 Drag& Drop API stellt keine spezifischen Ausdrucksmittel für Touchscreens bereit.)
Multimedia	+	Multimedia ohne proprietäre Plugins.
Textgliederung	+	Standard-Formatierung und Extraktion von Inhalten, z. B. für Inhaltsverzeichnisse. HTML5 als E-Book Format?
Microdata	+	Angebot typspezifischer Anwendungsfunktionen

Tab.: Datentabelle

HTML5

Hinsichtlich der Bezeichnung der aktuellen Version von HTML werden wir uns im weiteren Verlauf der Veranstaltung weitgehend der Diktion der WHATWG anschließen und auf den Gebrauch des Begriffs HTML5 verzichten. So erscheint die Betrachtung von HTML als *living standard* den tatsächlichen Verhältnissen weitaus angemessener als die Festlegung auf eine vermeintliche Versionsbezeichnung. Deren Bedeutung – d. h. der Umfang und die Form der darin inbegriffenen Ausdrucksmittel – ist ja nur bedingt stabil und wird vermutlich auch nach etwaiger Verabschiedung einer offiziellen Version weiterhin einem erheblichen Wandel unterliegen.

Eingeräumt werden soll jedoch auch, dass es sich bei HTML5 um ein durchaus wirksames Werbemittel handelt, da mit dieser Bezeichnung der Anspruch erhoben wird, dass mit der damit bezeichneten Technologie eine neue Generation von Webanwendungen ermöglicht wird. Diesbezüglich trifft die WHATWG in ihrem Spezifikationsdokument die folgende Aussage bezüglich der Frage, ob es sich hierbei um HTML5 handelt:



Hinweis

HTML5?

In short: Yes.

In more length: The term HTML5 is widely used as a buzzword to refer to modern Web technologies, many of which (though by no means all) are developed at the WHATWG, in some cases in conjunction with the W3C and IETF.

Dass es sich bei HTML5 jedoch um ein durchaus substantielles Buzzword handelt und der durch den Begriff suggerierte Generationswechsel nicht von der Hand zu weisen ist, sollte nicht zuletzt anhand der oben aufgezeigten Mehrwerte in Bezug auf die Entwicklung mobiler Anwendung deutlich geworden sein. Der martialische Charakter, der dem in der folgenden Abbildung dargestellten HTML5 Logo eignet, erscheint daher angesichts der tatsächlichen Überzeugungskraft der technologischen Ausdrucksmittel nur bedingt erforderlich, wenn auch mit Blick auf den Einsatz als Werbemittel tragbar.



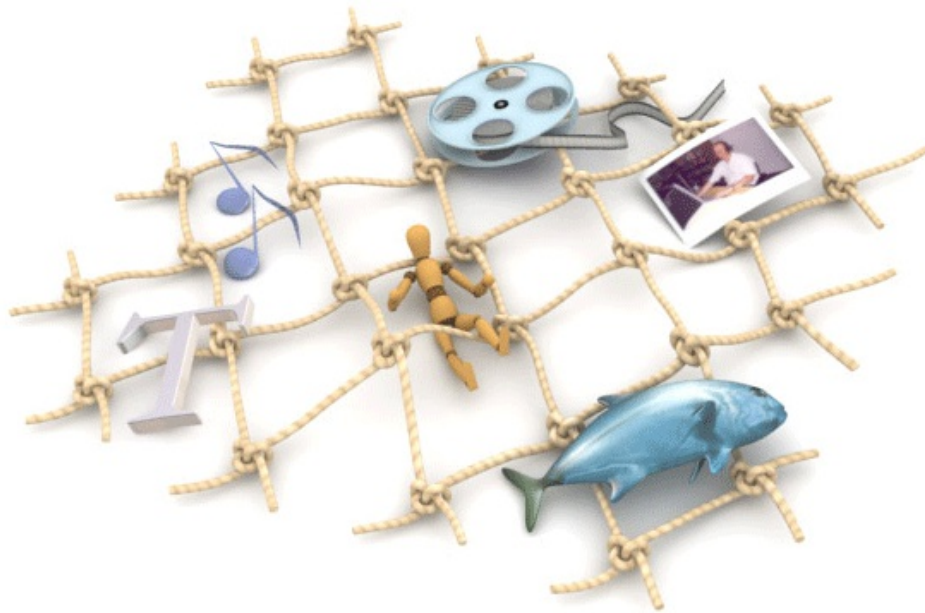
Abb.: HTML5 Logo

6 Struktur und Gestaltung

Die aktuellsten Ausdrucksmittel für HTML können in verschiedener Hinsicht als Anknüpfung und Weiterführung einiger Grundideen der Sprache angesehen werden. Dies gilt zum einen für die Multimedia-Elemente `<video>` und `<audio>`, auf die wir in der Lerneinheit „VMM - Verwendung von Multimedia“ detaillierter eingehen werden.

Diese verwirklichen die bereits in der Urfassung von HTML geäußerte Idee, derzufolge HTML als Auszeichnungssprache für Hypertext auch im Hinblick auf die Auslegung von Hypertext als Hypermedia betrachtet werden kann:

“HyperMedia is a term used for hypertext which is not constrained to be text: it can include graphics, video and sound.”



Ein anderer Schwerpunkt, den Sie hinsichtlich HTML aktuell erkennen können, betrifft das ebenfalls im Ursprung der Sprache enthaltene Bestreben, standardisierte Ausdrucksmittel für die semantische Auszeichnung der Inhalte von HTML-Dokumenten bereitzustellen. Diese Ausdrucksmittel werden wir nachfolgend vorstellen und ihre Anwendbarkeit mit Blick auf das von uns verwendete Gestaltungskonzept betrachten.

6.1 Semantisches Markup

Wie bereits erwähnt, stehen uns schon seit den ersten Versionen von HTML Markupelemente zur Auszeichnung von Inhalten als Überschrift (<h1>-<h6>), zur Hervorhebung von Inhalten () oder auch zur Markierung von einzelnen Abschnitten (<p>) als zusammenhängenden Sinneinheiten zur Verfügung. Zusätzlich dazu können wir nun weitere Aspekte textueller – oder in anderen Formen vorliegender – Inhalte hinsichtlich ihrer Zusammengehörigkeit, aber auch im Hinblick auf ihre Abgrenzung, markieren.

Ersterem dienen insbesondere die Elemente `article` und `section`, die denjenigen von Ihnen, die mit LaTeX vertraut sind, bekannt erscheinen dürften. Diese können eine Menge aufeinander bezogener oder ineinander eingebetteter Inhalte mittels `<section>` als zu einem gemeinsamen übergreifenden Thema gehörig kennzeichnen, welches mittels `<article>` markiert wird. Problematisch erscheint an diesen Elementen eventuell, dass Sie eine Analogiebildung zu „klassischen“ gegliederten und linear zu konsumierenden Texten aufweisen und nicht selbstverständlich auf andere Formen von Inhalten bezogen werden können.

Im folgenden werden wir daher aufzuzeigen versuchen, wie diese Elemente für die Umsetzung einer der Ansichten des Layoutkonzepts eingesetzt werden können, das wir einleitend vorgestellt haben. Beachten Sie aber, dass es sich hierbei nur um eine von mehreren `<article>` Möglichkeiten handelt, wie die betreffenden Elemente verwendet werden können.

`<article>` und `<section>` Wir beziehen uns dafür auf die folgende Ansicht, die einen Überblick über verschiedene Materialien darstellt, welche in unserer Anwendung zu einem bestimmten Werk vorhanden sind. Dabei kann es sich um Bilder, Filme und Audioinhalte handeln, um Textausschnitte aus Werken oder um andere Dokumente, die bezüglich des betreffenden Werks als „Zeitdokumente“ existieren. Außerdem ist es möglich, Querverweise zu anderen Inhalten – Werken, Personen oder Orten – herzustellen, für die gleichermaßen eine Ansicht wie die nachfolgend gezeigte aufgebaut werden kann. Schließlich existieren für ausgewählte Werke auch Einführungstexte, für die die Gestaltungsvorlage eine ausschnittsweise Anzeige in weißer Schrift auf blauem Hintergrund vorsieht:

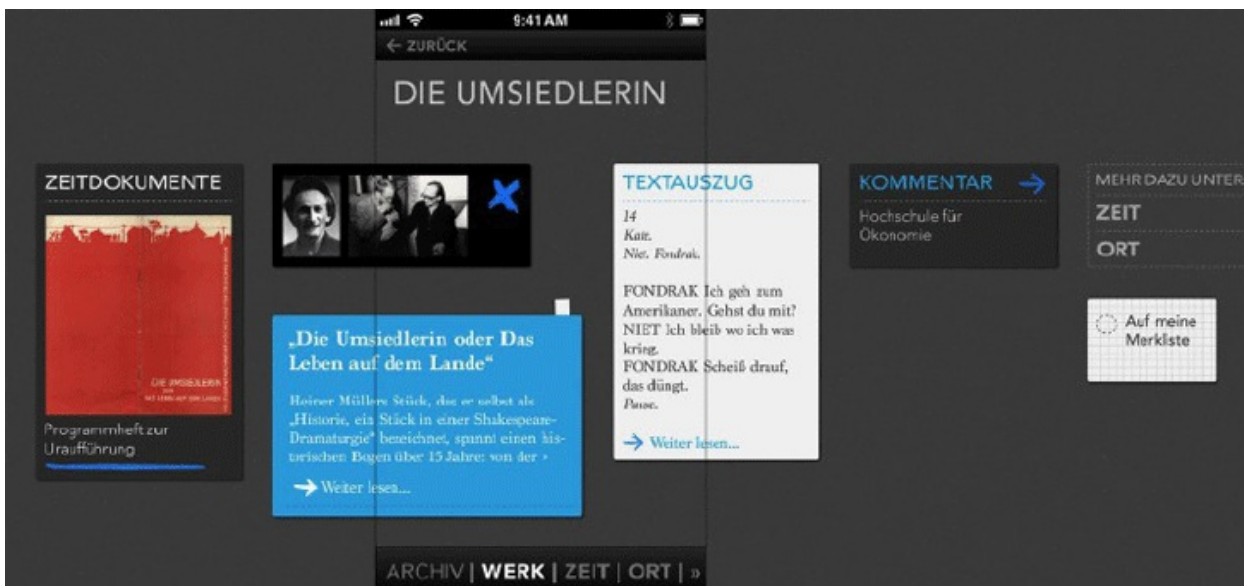


Abb.: Ansicht von `<article>` und `<section>`

Mit Blick auf die Semantik von `<article>` und `<section>` erscheint es uns denkbar, die hier gezeigte Gesamtansicht als `<article>` zu modellieren und die einzelnen Ansichtselemente, die Sie oben sehen, jeweils als `section` Elemente zu kennzeichnen – abzüglich der beiden Elemente am rechten Rand, bei denen es sich um Querverweise bezüglich des gesamten `<article>` bzw. um ein Aktionselement handelt, mittels dessen ein Nutzer das hier ausgewählte Werk einer „Merkliste“ hinzufügen kann. Die Grobstruktur des Markups könnte dann z. B. wie folgt aussehen:



Quellcode

<article> und <section> Elemente

```
001 <article>
002   <h1>Die Umsiedlerin</h1>
003   <section>
004     <h2>Zeitdokumente</h2>
005     <!-- (...) -->
006   </section>
007   <section>
008     <!-- (Bilder) -->
009     <!-- (...) -->
010   </section>
011   <section>
012     <!-- (Einführungstext) -->
013     <h2>Die Umsiedlerin oder das Leben auf dem Lande</h2>
014     <!-- (...) -->
015   </section>
016   <section>
017     <h2>Textauszug</h2>
018     <!-- (...) -->
019   </section>
020   <section>
021     <h2>Kommentar</h2>
022     <!-- (...) -->
023   </section>
024   <!-- (...) -->
025 </article>
```

Gliederungselemente

Sie sehen also, dass die betreffenden Elemente durchaus schlüssig auch auf eine Präsentationsform von Inhalten angewendet werden können, die zwar als gleichgeordnete Aspekte auf ein gemeinsames Thema – hier Heiner Müllers Stück „Die Umsiedlerin“ – bezogen werden können, jedoch nicht notwendigerweise in einer linearen Beziehung zueinander stehen, wie dies für die „Abschnitte“ eines „Artikels“ im landläufigen Sinne üblich ist.

Neben den beiden genannten Hauptgliederungselementen werden weitere Elemente eingeführt, mittels derer andere Aspekte der „Gewichtung“ von Inhalten in Bezug auf ein übergreifenden Thema ausgedrückt werden können. Dazu gehören u. a. die folgenden durchaus aussagekräftigen Elemente:

- `<details>` zur Kennzeichnung vertiefender/zusätzlicher Inhalte. Browser können vorsehen, dass die Darstellung der betreffenden Inhalte nach Nutzerinteraktion eingeblendet wird. Der Hinweis auf die Detailinhalte kann mittels eines `<summary>` Elements erfolgen, das nur im Kontext von `<details>` verwendet werden kann. Siehe dafür auch die [W3C Dokumentation des W3C](#).
- `<aside>` zur Kennzeichnung einer „Randbemerkung“ bzw. von Inhalten, die bei einem linearen Text außerhalb von dessen Textfluss stehen.

Unterhalb der hier beschriebenen Gliederungselemente können Sie weitere Elemente nutzen, um die Bezüge ggf. auch unterschiedlicher Inhaltstypen zueinander kenntlich zu machen. Beispielsweise erscheinen die Elemente `<figure>` und `<figcaption>` geeignet, um die interne Struktur des Gestaltungselement „Zeitdokumente“ zu modellieren:



Quellcode

<figure> und <figcaption> Elemente

```
001 <section>
002   <h2>Zeitdokumente</h2>
003   <figure>
004     
005     <figcaption>Programmheft der Uraufführung</figcaption>
006   </figure>
007 </section>
```

Querverweise

Beachten Sie auch hier, dass `<figure>` nicht ausschließlich zur Einbindung und Beschriftung von textuellen oder Standbildinhalten verwendet werden kann, sondern auch für die entsprechende Darstellung von Audio- oder Videoinhalten geeignet erscheint.

Auch den Bezugs von Querverweisen auf ein übergeordnetes Thema können wir mittels eines Strukturelements zum Ausdruck bringen. So steht uns für Querverweise selbst seit den Ursprüngen von HTML das `<a>` Element zur Verfügung, das wir z. B. für die Umsetzung des im Design vorgesehenen Gestaltungselements „*Mehr dazu unter...*“ verwenden können. Da letzteres im Gegensatz zu den anderen Elementen auf andere Themen verweist, die dem Thema „Die Umsiedlerin“ gleichgeordnet sind, erscheint uns für das Element selbst die Verwendung von `<aside>` angemessen:



Quellcode

`<aside>` Element

```
001 <article>
002   <h1>Die Umsiedlerin</h1>
003   <!-- (...) -->
004   <aside>
005     <ul>
006       <li><a href="(...)">Zeit</a></li>
007       <li><a href="(...)">Ort</a></li>
008     </ul>
009   </aside>
010   <!-- (...) -->
011 </article>
```

Falls ein Querverweis jedoch nicht wie hier eine genuin *inhaltliche* Funktion erfüllt, sondern dem Nutzer die Bedienung der Anwendung bzw. das „Umschalten“ zwischen verschiedenen Perspektiven ermöglichen soll, ist es nun möglich, diese „Navigationsfunktion“ durch Einbettung in ein `<nav>` Element kenntlich zu machen. So sieht das Anwendungsdesign beispielsweise die folgenden Kopf- und Fußleisten vor, die in allen Ansichten verfügbar sein sollen, um den Wechsel zwischen den Hauptansichten der Anwendung zu ermöglichen:



Abb.: Kopf- und Fußleiste

Deren Optionen könnten z. B. wie folgt in ein `<nav>` Element eingebettet werden:



Quellcode

`<nav>` Element

```
001 <!-- (...) -->
002 <nav>
003   <ul>
004     <li><a href="(...)">Archiv</a></li>
005     <li><a href="(...)">Werk</a></li>
006     <li><a href="(...)">Zeit</a></li>
007     <li><a href="(...)">Ort</a></li>
008   </ul>
009 </nav>
```

Kopf- und Fußzeile

Nicht zuletzt können Sie auch die Tatsache, dass die solche Optionen in einer Kopf bzw. Fußzeile angeboten werden, mittels der Elemente `<header>` und `<footer>` explizit kenntlich machen. Beachten Sie aber, dass diese Elemente im Gegensatz zu den vorstehend genannten einen klaren Bezug zum zweidimensionalen Aufbau einer Ansicht haben und damit nicht mehr als „reine Strukturelemente“ bezüglich der in einer Ansicht darzustellenden Inhalte angesehen werden können. Der Aufbau der Gesamtansicht eines ausgewählten Themas inklusive der Kopf- und Fußzeile könnte damit wie folgt realisiert werden:



Quellcode

<header> und <footer> Elemente

```
001 <body>
002   <header>
003     <a href="(...)">Zurück</a>
004   </header>
005   <article>
006     <h1>Die Umsiedlerin</h1>
007     <!-- (...) -->
008   </article>
009   <footer>
010     <nav>
011       <ul>
012         <li><a href="(...)">Archiv</a></li>
013         <!-- (...) -->
014       </ul>
015     </nav>
016   </footer>
017 </body>
```

Mehrwert der Strukturelemente

Der Mehrwert der neuen Strukturelemente von HTML wird offensichtlich, wenn Sie z. B. überlegen, wie Sie letztere Struktur ohne diese Elemente realisiert hätten. Vermutlich hätten Sie auf die Verwendung der in HTML vorgesehenen Elemente `<div>` und `` zurückgegriffen. Diese erlauben es uns, in Verbindung mit der Zuweisung eines eindeutigen Identifikators mittels des `id` Attributs oder der Zuweisung einer oder mehrerer Typen mittels `class` anwendungsspezifische Strukturelemente zu definieren, deren Funktion durch kein anderes HTML Element adäquat zum Ausdruck gebracht werden kann:



Quellcode

<div> Element


```
001 <body>
002   <div id="kopfzeile">
003     <a href="(...)">Zurück</a>
004   </div>
005   <div class="inhalt">
006     <h1>Die Umsiedlerin</h1>
007     <!-- (...) -->
008   </div>
009   <div id="fusszeile">
010     <div class="navigationsleiste">
011       <ul>
012         <li><a href="(...)">Archiv</a></li>
013         <!-- (...) -->
014       </ul>
015     </div>
016   </div>
017 </body>
```

Zweifelsohne ist für Sie als Leser die hier gezeigte Struktur in gleicher Weise verständlich und selbsterklärend wie diejenige, die wir zuvor auf Basis der standardisierten Strukturelemente umgesetzt haben. Weshalb sollten wir also auf die in ggf. langjähriger Entwicklungspraxis vertraut gewordene Verwendung von `<div>` und `` zugunsten der neuen Elemente verzichten? So ist deren Bedeutung ja, wie wir z. B. in der vorgeschlagenen Verwendung von `<section>` oder auch `<aside>` gesehen haben, ja ggf. sogar „abstrakter“ als anwendungsspezifische Bezeichner für `id` und `class`.

Stellen Sie sich aber einmal vor, dass Ihr Dokument nicht nur von Ihnen und den Browsern gelesen wird, in die es geladen wird, sondern auch von einer Suchmaschine, die die von Ihnen erstellten Inhalte interessierten Nutzern zugänglich machen soll. Von dieser Suchmaschine ist gewiss nicht zu erwarten, dass sie anwendungs- und sprachspezifische Bezeichner wie `fusszeile`, `inhalt` oder `navigationsleiste` „versteht“, wohl aber, dass sie ein „Verständnis“ bezüglich der Bedeutung der Elemente `<footer>`, `<article>` oder `<nav>` mitbringt. Natürlich geht dieses Verständnis ggf. darauf zurück, dass die Entwickler der Suchmaschine in Kenntnis der im Standard festgelegten Bedeutung dieser Elemente eine spezifische Behandlung dieser explizite Elemente vorsehen – für beliebige sprachspezifische `class` Bezeichner ist dies nicht zu erwarten.

Dieser Versuch einer „Einführung in eine Suchmaschine“ bzw. deren Entwickler zeigt, was im Kern die Motivation der Strukturelemente von HTML ist. Mittels ihrer soll es möglich sein, Transparenz bezüglich der Inhalte eines Dokuments auf allgemein verständliche Weise – und eben nicht anwendungsspezifisch – herzustellen, um eine höhere Präzision von Suchanfragen zu ermöglichen. So macht beispielsweise die Verwendung von `<figcaption>` deutlich, dass der textuelle Inhalt dieses Elements eine Beschreibung der Inhalte ist, die in der umgebenden `<figure>` enthalten sind. Handelt es sich beispielsweise um ein Bild, Video oder Audio, weist `<caption>` darauf hin, was dort zu sehen bzw. zu hören ist.

Für eine Suche über Multimediainhalten liefert das Vorkommen eines Suchbegriffs in einem `<figcaption>` Element damit deutliche Anhaltspunkte dafür, dass der betreffende Inhalt der `<figure>` tatsächlich relevant aus der Sicht des Suchenden ist. In gleicher Weise kann die Relevanz eines Dokuments bezüglich eines Suchbegriffs in Abhängigkeit davon unterschiedlich beurteilt werden, ob dieser Begriff z. B. innerhalb eines `<details>` oder eines `<aside>` Begriffs auftritt.

Im Hinblick auf diese Einführung neuer Strukturelemente bewegt sich HTML damit nahe an seinen Ursprüngen und denen des WWW. So wurde dieses ja  ursprünglich eingeführt als *“(...) a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents.”* Als weiterer relevanter Aspekt, der mit der Einführung standardisierter Strukturelemente und der dadurch erhöhten Selbsterklärungsfähigkeit von Dokumenten einhergehen, sei an dieser Stelle die Barrierefreiheit erwähnt, die z. B. durch bessere Unterstützung von automatischen Screenreadern für Sehbehinderte daraus resultiert. Nicht zuletzt liefern die durch Strukturelemente ermöglichten Relevanzaussagen auch einen Anhaltspunkt dafür, wie die Inhalte eines Dokuments für die Anzeige auf einem Gerät mit beschränkten Darstellungskapazitäten ggf. gefiltert werden können oder welche Alternativen bezüglich der Anordnung der Elemente bestehen.

Die Verfügbarkeit der HTML-Strukturelemente bedeutet jedoch nur eine Lösung für diejenigen strukturellen Aspekte von Dokumenteninhalten, die tatsächlich anwendungsübergreifend von Bedeutung sind. Das Ende der Verwendung anwendungsspezifischen Markups mittels `<div>` und `` wird hiermit keineswegs eingeläutet. Der nachfolgende Abschnitt wird nun bezüglich der Verwendung dieser Elemente einige Grundlinien benennen, die im Hinblick auf die Darstellung von HTML-Dokumenten mittels CSS zu beachten sind.

6.2 Anwendungsspezifische Markupelemente

Unabhängig davon, welche konkreten anwendungsspezifischen oder generischen Ausdrucksmittel Sie für den Aufbau eines HTML Dokuments verwenden, spielt die zu realisierende Gestaltungsvorlage – bzw. die Menge der möglichen Darstellungsformen eines Dokuments auf ggf. unterschiedlichen Endgeräten – eine wesentliche Rolle bei der Entscheidung bezüglich der zu wählenden Dokumentenstruktur. Dies schließt auch Anforderungen bezüglich der Interaktion mit den dargestellten Inhalten ein. So existiert in Bezug auf die oben gezeigte Gestaltungsvorlage beispielsweise die Anforderung, dass die verschiedenen Bildelemente als Block horizontal scrollbar sein sollen, ohne dass die vorgesehene Überschrift beim Scrollen verschoben wird.

Diese Anforderung macht eine zusätzliche darstellungsbezogene Strukturierung der Inhalte erforderlich, welche nicht auf deren semantische Struktur zurückgeführt werden kann, die wir mittels des bisher vorgeschlagenen Markups und des Einsatzes der standardisierten Strukturelemente zu erfassen versuchten. An dieser Stelle liegt also eine klare Indikation für den Einsatz eines anwendungsspezifischen Strukturelements vor, für das wir ein `<div>` Element mit geeigneter `class` Markierung verwenden. Freilich sei angemerkt, dass die vorgeschlagene Markierung durchaus anwendungsübergreifend geeignet erscheint und die Funktion einer scrollbaren Teilansicht z. B. in nativen mobilen Anwendungen durch entsprechende Ansichtselemente unterstützt wird.



Quellcode

Ansichtselemente

```
001 <article>
002   <h1>Die Umsiedlerin</h1>
003   <!-- scrollable area -->
004   <div class="scrollview">
005     <section>
006       <h2>Zeitdokumente</h2>
007       <!-- (...) -->
008     </section>
009     <section>
010       <!-- (Bilder) -->
011       <!-- (...) -->
012     </section>
013     <!-- (...) -->
014   </div>
015   <!-- end of scrollable area -->
016 </article>
```

Eine solche Verwendung von `<div>` Elementen ist immer dann erforderlich, wenn eine *Menge* von Markupelementen als *ein* Gestaltungselement betrachtet werden muss, ohne dass auf Ebene des Markups bereits ein Markupelement existiert, dem die erforderlichen Gestaltungseigenschaften auf Ebene von CSS zugewiesen werden könnten. Dies gilt z. B. auch für die Zuweisung von relativen oder absoluten Gesamtgrößen zu einer Menge von Markupelementen, die zunächst nur als Schwesterelemente vorliegen.

Ein anderer Fall, bei dem der Einsatz anwendungsspezifischen Markups erforderlich erscheint, liegt dann vor, wenn sich aus einer Gestaltungsvorlage wiederkehrende Gestaltungseinheiten ablesen lassen, für die eine verallgemeinerte Zuweisung von Stileigenschaften wünschenswert ist. In unserer Designvorlage ist dies beispielsweise für den internen Aufbau des Gestaltungselements „Textauszug“ der Fall. Konkret haben wir es hier mit einem Dramentext zu tun, der im Kern aus einer Abfolge von Dialogschritten der jeweils auftretenden Akteure besteht, dem ggf. Regieanweisungen vorangestellt oder nachgestellt werden können. Das vorgeschlagene Design macht diesen Aufbau anhand von Unterschieden im Schriftschnitt und Normal- vs. Großschreibung transparent:

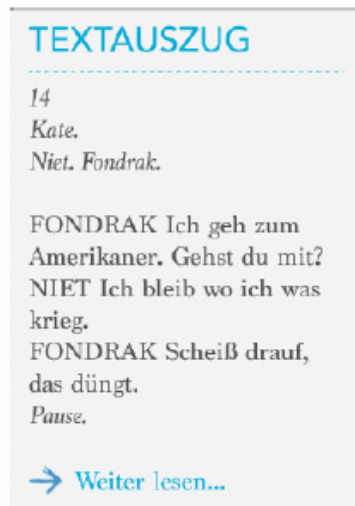


Abb.: Beispiel Textauszug

Für die Umsetzung bestehen hinsichtlich der Strukturierung eines HTML Dokuments grundsätzlich mehrere Möglichkeiten. So wäre es denkbar, `<div>` oder `` Elemente mit darstellungsbezogenen Attributen zu versehen, anhand derer auf der Ebene von CSS die Formatierung der einzelnen Textsegmente wie im Designdokument vorgesehen zugewiesen werden kann.

Würden für die Realisierung der Kursivschrift in den Regieanweisungen oder der Großschreibung der Akteursnamen aber z. B. Attributsetzungen wie `class="italics"` or `class="capitals"` verwendet, so würde damit zwar der Anforderung Rechnung getragen, dass HTML frei von darstellungsbezogenen Elementen wie `<i>` oder `` bleiben sollte, um die Formatzuweisung allein auf Ebene von CSS durchzuführen. Eine in diesem Maße „darstellungsbezogene Semantik“ der anwendungsspezifischen Markupelemente würde jedoch einen Formatierungsspielraum nur um den Preis eines Verlusts an Aussagekraft dieser Elemente eröffnen – so erschiene es zumindest fragwürdig, in CSS einem Element mit Attribut `class="capitals"` keine Großschrift, sondern Fettschrift als Alternative zuzuweisen.

Als Alternative bietet es sich daher an, für anwendungsspezifisches Markup ein „echtes“ semantisches Markup zu verwenden, das die darzustellenden Inhalte im Hinblick auf ihre *Bedeutung* im gegebenen Anwendungsbereich kennzeichnet. D. h. es sollte jeweils zum Ausdruck bringen, was für ein Typ von Daten im Hinblick auf ein anwendungsspezifisches Datenmodell vorliegt. Für das genannte Beispiel wäre z. B. das folgende Markup denkbar:



Quellcode

Beispiel für semantisches Markup

```
001 <section>
002   <h2>Textauszug</h2>
003   <div class="szene">
004     <div class="regieanweisungen">
005       <!-- (...) -->
006     </div>
007     <div class="dialogschrift">
008       <span class="akteur">Fondrak<span>
009       <span class="aeusserung">Ich geh zum Amerikaner. Gehst Du mit.</span>
010     </div>
011     <!-- (...) -->
012     <div class="regieanweisungen">
013       Pause
014     </div>
015   </div>
016   <!-- (...) -->
017 </section>
```

Dieses Beispiel zeigt, dass eine gelungene Gestaltungsvorlage bereits größtenteils eine Assoziation der semantischen Struktur von Inhalten und der gestalterischen Struktur der Darstellung dieser Inhalte vornimmt. So hatten wir die hier verwendeten anwendungsspezifischen Inhaltstypen – *regieanweisungen*, *dialogschrift*, *akteur*, *etc.* – aus der Gestaltungsvorlage unter Aufbietung unseres Wissens über die Konzepte von Dramentexten „herausgelesen“. In gleicher Weise ist anzunehmen, dass die Designerin ihre Kenntnis dieser Konzepte und von deren Struktur in die vorgeschlagene Gestaltung eingebracht hat. Auf diese Weise existieren in vielen Fällen für einfache und komplexe Sinneinheiten auf der *Inhaltsebene* entsprechende einfache bzw. komplexe Gestaltungseinheiten auf der *Darstellungsebene* und umgekehrt.

Steht uns für die Repräsentation einer Sinneinheit jedoch kein geeignetes Strukturelement zur Verfügung, dann ist es unumgänglich, hierfür ein geeignetes anwendungsspezifisches Element mittels `<div>` oder `` einzuführen. Um möglichst flexibel im Hinblick auf Gestaltungsalternativen zu sein, empfiehlt sich hier eine semantische Auszeichnung, die den betreffenden Inhaltstyp der Sinneinheit im Rahmen der Domäne, d. h. des inhaltlichen Anwendungsbereichs, der betreffenden Anwendung identifiziert. Im vorliegenden Fall stellen literarische Texte und deren Strukturierung diesen Anwendungsbereich dar.

Freilich ist auch ein solcher Anwendungsbereich nicht spezifisch für eine bestimmte Anwendung, sondern verallgemeinerbar. Die Einführung anwendungsspezifischer Bezeichner wie *regieanweisungen*, *akteur*, *etc.* zum Zweck der Identifikation der Inhaltstypen erscheint damit zunächst gleichermaßen fragwürdig – weil anwendungsspezifisch – wie die oben verworfene Verwendung solcher Bezeichner zur Kennzeichnung von Strukturmerkmalen wie *kopfzeile*, *fusszeile*, *etc.*



Abb.: Illustration des W3C zur Illustration der Motivation für RDFa

© W3C [RDFa 1.1 Primer - Second Edition](#)

Die originale Bildunterschrift lautet: "On the left, what browsers see. On the right, what humans see. Can we bridge the gap so that browsers see more of what we see?"

Im Gegensatz zu unseren gerade vorgeschlagenen Bezeichnern für anwendungsunabhängige Inhaltstypen handelt es sich bei den mittels `<header>`, `<article>` *etc.* auszuzeichnenden Strukturmerkmalen jedoch um domänenunabhängige Merkmale, die durch eine in Bezug auf darzustellende Inhalte *domänenunabhängige* Markupsprache wie HTML grundsätzlich abgedeckt werden können. Die Einführung anwendungsübergreifender, aber domänenspezifischer Ausdrucksmittel liegt damit außerhalb dessen, was durch HTML als Standard geleistet werden kann. Allerdings wird im Rahmen der laufenden Standardisierungsdiskussion durchaus die Möglichkeit in Betracht gezogen, Inhalte von HTML Dokumenten mit domänenspezifischen Auszeichnungen zu versehen, die anwendungsunabhängig definierbar sind. Dazu gehört das oben erwähnte als [WWW](#) Microformats bezeichnete Auszeichnungsformat sowie das als Alternative dazu diskutierte Format [WWW](#) RDFa.

Dessen Motivation wird anhand eines Beispiels illustriert, das dem von uns hier verwendeten sehr nahe ist und das wir in der vorigen Abbildung „Illustration des W3C“ als Abrundung unserer Ausführungen zitieren. Die dort ebenfalls zitierte Bildunterschrift macht jedoch deutlich, dass es bei RDFa nicht nur darum geht, Anwendungsentwickler bei der Findung von `class`-Bezeichnern für anwendungsspezifisches Markup durch Bereitstellung anwendungsunabhängig definierter domänenspezifischer Bezeichner zu entlasten.

Eine solche Entlastung ist allenfalls ein willkommener Nebeneffekt der Transparenz, welche HTML Dokumenten bei Anwendung von RDFa hinsichtlich ihrer Inhalte verliehen würde. Die auf diese Weise erreichte sprachunabhängige Selbsterklärungsfähigkeit der Inhalte eines Dokuments für maschinelle Zugriffe würde denn auch deutlich über die Transparenz hinaus gehen, die durch Anwendung der aktuell standardisierten rein *strukturellen* Markupelemente erzielt werden kann.

Zusammenfassung

- Auszeichnungssprachen lassen sich als Sprachen auffassen, die das Hinzufügen von zusätzlichen Inhalten zu bestehenden Inhalten bei Gewährleistung der Unterscheidbarkeit beider Typen von Inhalten ermöglichen.
- Maschinenlesbare Auszeichnungssprachen unterscheiden sich hinsichtlich des Umfangs an sprachspezifischen Symbolen und dadurch, wie diese Symbole zum Zweck der Auszeichnung angewendet werden.
- XML ist keine *konkrete* Auszeichnungssprache. XML definiert generische Ausdrucksmittel für die Repräsentation von Auszeichnungsinhalten in Textdokumenten, die zum einen die Unterscheidbarkeit von auszeichnenden und ausgezeichneten Inhalten, zum anderen die Zuordnung der Inhalte zueinander ermöglichen.
- In seiner Urfassung erscheint HTML als eine Markupsprache für Hypertext, die mittels geeigneter Elemente die Eigenschaften von Textdokumenten und die Verknüpfung von Texten zu beschreiben ermöglicht.
- Im Gegensatz zu XML ist HTML eine konkrete Auszeichnungssprache.
- Die Standardisierung der Ausdrucksmittel für JavaScript in HTML reduziert die Notwendigkeit, proprietäre ggf. browserspezifische Skriptlösungen zu implementieren und erlauben es, proprietäre Plugins durch browserunterstützte Funktionen abzulösen.
- Strukturelemente von HTML ermöglichen es, Transparenz bezüglich der Inhalte eines Dokuments auf allgemein verständliche Weise – und eben nicht anwendungsspezifisch – herzustellen. Hierdurch entsteht eine höhere Präzision von Suchanfragen.

Sie sind am Ende dieser Lerneinheit angelangt. Auf den folgenden Seiten finden Sie noch Übungen.

Übungen

Zur Bearbeitung der Übungen empfehlen wir Ihnen die Installation der Aptana IDE.

Installationsanleitung für Aptana

Die Implementierungsbeispiele für die vorliegende Lerneinheit finden Sie im Projekt `org.dieschnittstelle.iam.htm`. Die kompletten Beispiele können Sie hier herunterladen:

 iam_bispiele_komplett.zip (45 MB)

Aus urheberrechtlichen Gründen enthalten die Implementierungsbeispiele andere Abbildungen als im Lehrmaterial gezeigt. Bitte beachten Sie stets das Copyright und stellen Sie sicher, dass geschützte Abbildungen nicht frei über das Internet verfügbar werden - bspw. über ein GIT Repository.



Programmieren

Übung HTM-01

Inbetriebnahme der Beispiele

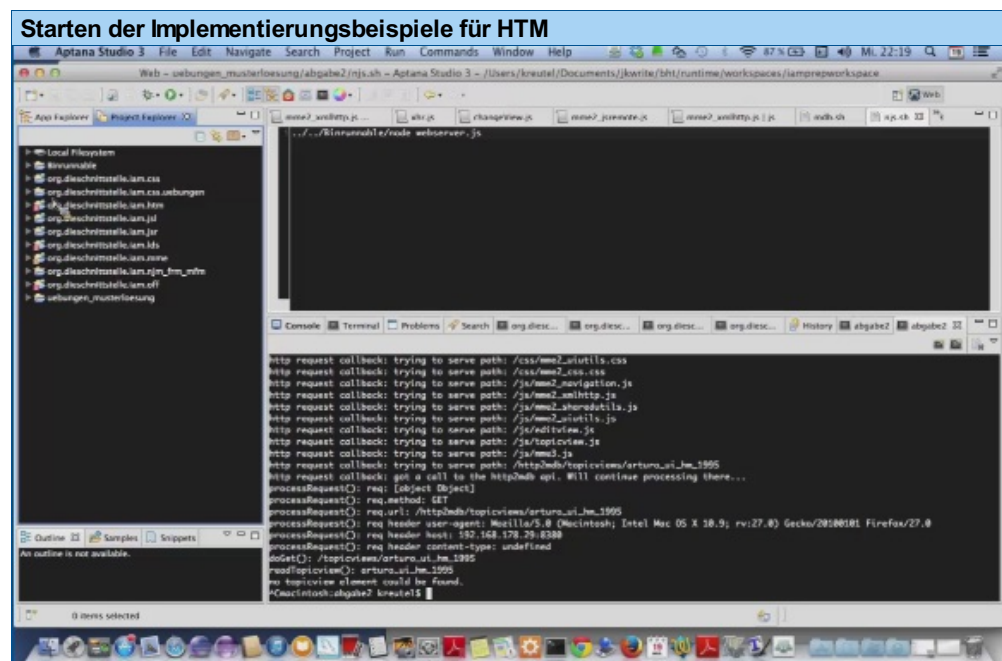
1. Laden Sie die Implementierungsbeispiele herunter.
2. Öffnen Sie Aptana und importieren Sie die Beispiele (Import... → General → Existing Projects into Workspace → Select Archive File) in Ihren Workspace.
3. Überprüfen Sie die Server-Einstellungen in Aptana (Einstellungen → Aptana Studie → Web Servers → 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.

Bearbeitungszeit: 30 Minuten

Im folgenden Film sehen Sie wie Sie die Implementierungsbeispiele starten.



Film



© Beuth Hochschule Berlin - Dauer: 01:25 Min. - Streaming Media 3 MB



Programmieren

Übung HTM-02

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.

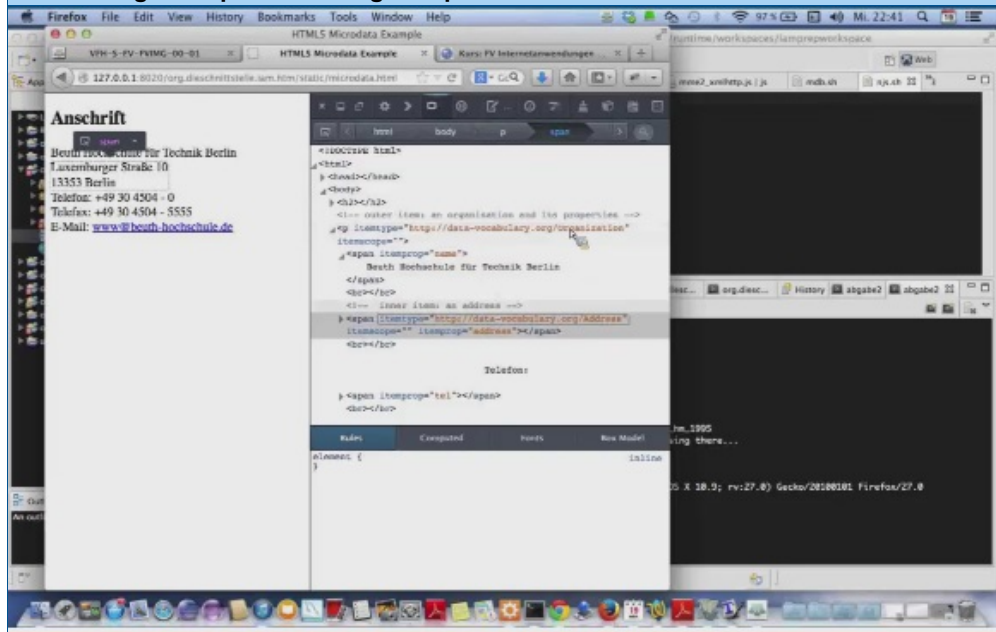
Bearbeitungszeit: 30 Minuten

Im folgenden Film stellt der Autor die Implementierungsbeispiele für die Lerneinheit HTM vor.



Film

Vorstellung der Implementierungsbeispiele zu HTM



© Beuth Hochschule Berlin - Dauer: 06:49 Min. - Streaming Media 12 MB



Übung HTM-03

Dokumentenstruktur

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 Übung CSS03 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 Verknüpfungen *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?

Bearbeitungszeit: 90 Minuten

Wissensüberprüfung

Versuchen die hier aufgeführten Fragen zu den Inhalten der Lerneinheit selbständig kurz zu beantworten, bzw. zu skizzieren. Wenn Sie eine Frage noch nicht beantworten können kehren Sie noch einmal auf die entsprechende Seite in der Lerneinheit zurück und versuchen sich die Lösung zu erarbeiten.



Formulieren

Übung HTM-04

Auszeichnungssprachen

Versuchen die hier aufgeführten Fragen selbständig kurz zu beantworten, bzw. zu skizzieren.

1. Wozu dienen Auszeichnungssprachen im allgemeinen?
2. Welche wesentliche Unterscheidbarkeit muss bei Anwendung einer Auszeichnungssprache gewährleistet sein?
3. Wie kann die Unterscheidbarkeit bei maschinenlesbaren Auszeichnungssprachen gewährleistet werden?
4. Worin unterscheiden sich Auszeichnungssprachen außer in ihrer Syntax?
5. Welche Typen von Auszeichnungssprachen lassen sich im allgemeinen voneinander unterscheiden?

Bearbeitungszeit: 20 Minuten



Formulieren

Übung HTM-05

XML

Versuchen die hier aufgeführten Fragen selbständig kurz zu beantworten, bzw. zu skizzieren.

1. Inwiefern ist die XML eine Auszeichnungssprache?
2. Welche Struktur haben alle XML Dokumente gemeinsam?
3. Nennen Sie vier syntaktische Regeln der XML.
4. Erläutern Sie die Konzepte von Wohlgeformtheit und Gültigkeit eines Textdokuments im Sinne der XML.
5. Warum kann es wichtig sein, dass die Gültigkeit eines Markupsprachen-Dokuments verifiziert werden kann?

Bearbeitungszeit: 20 Minuten



Formulieren

Übung HTM-06

HTML

Versuchen die hier aufgeführten Fragen selbständig kurz zu beantworten, bzw. zu skizzieren.

1. Entspricht HTML den Anforderungen der XML?
2. Nennen Sie drei mögliche Abweichungen zwischen HTML und den Regeln der XML.
3. Weshalb hat sich XML konformes HTML nicht flächendeckend durchgesetzt?
4. Was ist ein Defizit von HTML, das auf dessen dokumentenbezogene Sichtweise zurückgeführt werden kann?
5. Was ist der Anwendungsbereich/ die Domäne von HTML?
6. Auf welcher Grundlage werden die Referenzen auf HTML Dokumente und andere Inhalte, die ein HTML Dokument enthalten kann, aufgelöst?

Bearbeitungszeit: 20 Minuten



Formulieren

Übung HTM-07

HTML5

Versuchen die hier aufgeführten Fragen selbständig kurz zu beantworten, bzw. zu skizzieren.

1. Inwiefern wird mit dem Begriff 'HTML5' eine Auszeichnungssprache bezeichnet und was bezeichnet der Begriff im weiteren Sinne?
2. Nennen Sie 5 Ausdrucksmittel, die mit der Bezeichnung 'HTML5' gemeint sein können.
3. Nennen Sie 4 kommerzielle Mehrwerte von HTML5.
4. Nennen Sie 5 Elemente, die HTML5 zur Strukturierung von Inhalten neu einführt.
5. Inwiefern dient HTML5 der Strukturierung von Multimedia-Inhalten?
6. Was sind technische Vorteile der semantischen Markup-Elemente von HTML5?
7. Was sind Vorteile des semantischen Markups aus Nutzersicht?

Bearbeitungszeit: 30 Minuten



Formulieren

Übung HTM-08

Dokumentstruktur und Gestaltung

Versuchen die hier aufgeführten Fragen selbständig kurz zu beantworten, bzw. zu skizzieren.














1. Wovon hängt die Entscheidung für eine konkrete Dokumentenstruktur ab?
2. Welcher Zusammenhang besteht zwischen Sinneinheiten/Bedeutungseinheiten und Gestaltungseinheiten?
3. Welche 4 Ausdrucksmittel stellt HTML für die Formulierung anwendungsspezifischer semantischen Markups zur Verfügung?

Bearbeitungszeit: 10 Minuten

Übung HTM-09

Korrekturzeichen

Bitte ordnen Sie die Bezeichnungen den richtigen Korrekturzeichen zu.

<input type="text"/>	I FL	<input type="text"/>	
<input type="text"/>	Π	<input type="text"/>	
<input type="text"/>	H	<input type="text"/>	
<input type="text"/>	J	<input type="text"/>	
<input type="text"/>	L	<input type="text"/>	
<input type="text"/>	C	<input type="text"/>	
<input type="text"/>		<input type="text"/>	
<input type="text"/>		<input type="text"/>	
<input type="text"/>		<input type="text"/>	
<input type="text"/>	=	<input type="text"/>	

Wortabstand zu groß	Wortreihenfolge	Wort zusammen	Einzug
falsches Wort	kursiv	Wortabstand zu klein	kein Einzug
Deletur-Zeichen	Zeilenabstand zu klein	Absatz	fehlende Zeile
Zeilenabstand zu groß	Zeichendreher	Wortzwischenraum	falsches Zeichen
Blockade-Zeichen	Korrektur rückgängig	Zwiebelfisch	Absätze verbinden

? Test wiederholen Test auswerten

Appendix

Installation der Aptana IDE

1. Laden Sie sich von der [www. Aptana Webseite](http://www.apтана.com/) die aktuelle Version von Aptana Studio als Standalone Version für Ihre Rechnerplattform herunter und installieren Sie diese.
2. Installieren Sie ebenfalls die aktuelle Version von Firefox von [www. http://www.mozilla.org/en-US/firefox/fx/#desktop](http://www.mozilla.org/en-US/firefox/fx/#desktop).
3. Öffnen Sie Firefox und installieren Sie das Firebug Add-on von [www. https://getfirebug.com/](https://getfirebug.com/).

Erstellen Sie zu Beginn eine neue Webanwendung

1. Starten Sie Aptana
2. Wählen Sie `file/New/Web Project` bzw. `file/New/Other/Web/Web Project`.
3. Wählen Sie `Next` und dann die Option `Basic Web Template` aus.
4. Wählen Sie einen Namen Ihrer Wahl.
5. Schließen Sie den Vorgang mit `Finish` ab.

Starten Sie die Anwendung

1. Rufen Sie `Run as/JavaScript Web Application` aus dem Kontextmenü der `index.html` Datei im Wurzelverzeichnis der Anwendung auf.
2. Wenn Firefox als Default-Browser gesetzt ist, dann sollte jetzt die `index.html` Seite in Firefox geöffnet werden. Greifen Sie andernfalls in Firefox auf die URL `http://127.0.0.1:8020/$<$projektname>$/index.html`, wobei `<projektname>` der von Ihnen gewählte Applikationsname ist.

Nutzen Sie Firebug

1. Öffnen Sie Firebug mittels Auswahl von `View/Firebug` in der Firefox Menüleiste
 2. Wählen Sie in Firebug den Karteireiter `Net`.
 3. Laden Sie die `index.html` Seite neu (refresh).
 4. Nun sollte Ihnen in Firebug ein `GET Http Request` angezeigt werden, den Sie durch „Aufklappen“ inspizieren können.
 5. Mehr zu den Ausdrucksmitteln von HTTP erfahren Sie in der nächsten Lerneinheit.
-