HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

BACHELORARBEIT

Herr
Philipp Bammes

Barrierefreiheit im Internet

Untersuchung zur Erbringung von Barrierefreiheit unter Verwendung von HTML5

2014

Fakultät Medien

BACHELORARBEIT

Barrierefreiheit im Internet

Untersuchung zur Erbringung von Barrierefreiheit unter Verwendung von HTML5

Autor:
Herr Philipp Bammes

Studiengang:
Medientechnik

Seminargruppe:
MT11wD-B

Erstprüfer:
Prof. Dr.-Ing. Robert J. Wierzbicki

Zweitprüfer:
Dipl.-Ing. Jens Schuppe

Einreichung:
Mittweida, 24.06.2014

Faculty of Media

BACHELOR THESIS

Accessibility on the Internet

Examination of providing accessibility using HTML5

author:
Mr Philipp Bammes

course of studies:
Media Technology

seminar group:
MT11wD-B

first examiner:
Prof. Dr.-Ing. Robert J. Wierzbicki

second examiner:
Dipl.-Ing. Jens Schuppe

submission:
Mittweida, 24.06.2014

Bibliografische Angaben:

Bammes, Philipp

Barrierefreiheit im Internet

Untersuchung zur Erbringung von Barrierefreiheit unter Verwendung von HTML5

Accessibility on the Internet

Examination of providing accessibility using HTML5

2014 - 55 Seiten

Mittweida, Hochschule Mittweida (FH), University of Applied Sciences,

Fakultät Medien, Bachelorarbeit, 2014

Abstract

Diese Bachelorarbeit untersucht, wie Barrierefreiheit im Internet insbesondere unter Verwendung von HTML5 realisiert werden kann. Der Fokus liegt dabei auf der Beschreibung nativer Webtechnologien, die Drittanbieter-Plugins auf Webseiten überflüssig machen können und die sich somit positiv auf die Barrierefreiheit auswirken, sowie auf der Erläuterung inhaltsspezifischer HTML5-Elemente zur semantischen Strukturierung von Webseiten. Im Anschluss wird untersucht, welche technologischen Möglichkeiten es auf Seiten der Webentwickler gibt, gehörlose Menschen stärker ins Internet zu inkludieren, da die Arbeit zeigt, dass in diesem Bereich bislang keine nennenswerten Erfolge erzielt wurden. Der letzte Teil beschreibt die Konzeption eines Übersetzungs-Programms, das in der Lage ist, geschriebene Worte in Gebärden darzustellen. Damit wird die technische Umsetzbarkeit eines solchen Werkzeugs demonstriert.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Am Anfang war das Web.
Und das Web war aus Text.
Und Text war barrierefrei.

Zu Weihnachten des Jahres 1990 startete Tim Berners-Lee am schweizerischen CERN[1] den weltweit ersten Server, den „CERN httpd“ und legte damit den Grundstein für den globalen Siegeszug des World Wide Webs (WWW). Die ebenfalls von Berners-Lee entwickelte Sprache „HTML“ (Hypertext Markup Language) diente dabei als Textauszeichnungssprache zur einfachen Kommunikation zwischen dem eine Webseite ausliefernden Server und dem die Seite darstellenden Browser des anfragenden Clients. In den darauffolgenden Jahren verbreitete sich das World Wide Web rasant über den gesamten Globus und bildete schon bald die Grundlage für einen Großteil des weltweiten Informationsaustausch.

Da die erste, ursprüngliche Version von HTML rein textbasiert war, schnell jedoch Bestrebungen aufkamen, Bilder und später auch Videos im World Wide Web darzustellen, begann man mit der Erweiterung der Sprache, mit dem Ziel, Webseiten grafisch und damit ansprechender gestalten zu können. So entstanden mit der gleichzeitig stattfindenden Verbesserung der existierenden Browser leistungsstärkere Versionen von HTML. Ein wichtiger Meilenstein war ab 1994 zudem die Entwicklung der Layout-Sprache „CSS“ (Cascading Style Sheets), die die übliche Praxis, rudimentäre Gestaltungen von Seiten direkt im HTML vorzunehmen[2], durch die Auslagerung von Layout-Angaben langsam ablöste. Für multimediale Inhalte wiederum wurde in der explosionsartigen Verbreitung neuer, eigenständiger Techniken, die mittels Plugins in den Browsern angezeigt werden konnten, das „Flash“-Format der Firma „Adobe“ besonders populär und verbreiteter als alle anderen Techniken, mit dem sich beliebige Layouts, Animationen und Benutzerführungen auf Webseiten umsetzen ließen. Auch für die Verwendung von Videos auf Webseiten war das Format unerlässlich. Flash-Animationen und flashbasierte Webseiten dominierten das World Wide Web für mehr als 15 Jahre. Erst mit der allmählich zunehmenden Verbreitung der fünften Version von HTML, die unter ande-

1 Das CERN (Europäische Organisation für Kernforschung) ist eine Einrichtung zur physikalischen Grundlagenforschung.

2 Weit verbreitet waren insbesondere HTML-Tabellen, die ein Gestaltungsraster auch ohne CSS ermöglichten.

rem eine native Darstellung[3] von Audio- und Video-Dateien ermöglicht, begann der Rückgang von Flash-basierten Internetauftritten.

Gleichzeitig mit der Verbreitung des World Wide Webs begann auch die Entwicklung von Geräten und Schnittstellen, die körperlich eingeschränkten Menschen helfen sollten, die vom Browser dargestellten Informationen zu konsumieren. Besonders hervorzuheben sind dabei Sprachausgabe-Systeme, die für blinde Menschen den Textteil einer Webseite entweder mit einer computerbasierten Stimme vorlesen[4], oder mittels einer sog. „Braille-Zeile“ in tastbare Braille-Zeichen umwandeln.

Diese Systeme haben jedoch einen großen Nachteil, der sich aus dem technischen Aufbau der zu übersetzenden Webseite ergibt: Sie sind nicht in der Lage, andere Informationen als Text (und Bildbeschreibungen) aus einer Seite zu extrahieren und zu übersetzen. Dies führt zu einer paradoxen Situation in der Entwicklung des World Wide Web:

Während in den Zeiten hauptsächlich textbasierter Webseiten Hilfssysteme für körperlich beeinträchtigte Menschen ohne größere Probleme den Inhalt einer Seite erkennen und ihn in einer für den Anwender sinnvollen Reihenfolge und mit inhaltlichen Abgrenzungen vortragen konnten, präsentierten sich dieselben Seiten für sehende Nutzer oftmals wenig ansprechend und eintönig. Mit der oben erwähnten Einführung und großflächigen Verbreitung neuer Techniken (insbesondere „Flash“) verbesserte sich das Aussehen der Seiten, sie wurden also für sehende Nutzer attraktiver. Gleichzeitig verschlechterte sich die Lesbarkeit für Übersetzungs-Systeme immens: Grund dafür ist die in sich abgeschlossene Container-Struktur von Flash und anderen Plugin-Formaten, was dazu führt, dass die Inhalte nicht in maschinenlesbarer Form vorliegen. Besonders Webseiten, die komplett Flash-basiert sind, verschließen sich eingeschränkten Menschen vollständig. Verbesserung der Benutzeroberfläche für „gesunde“ Menschen führte also zu Verschlechterung der Inklusion körperlich behinderter Menschen.

Um diesem Trend entgegen zu wirken, gründeten sich Initiativen zur Erhaltung und Verbesserung der Erreichbarkeit von Internetseiten für eingeschränkte Menschen.

3 „nativ“ bedeutet in der EDV, dass Systeme eine bestimmte Funktionalität ohne Verwendung von Emulatoren, Plug-ins etc. bereitstellen kann.

4 Solche Systeme werden in der Regel als „Screenreader“ bezeichnet.

1.1 Problemstellung

Diese Bestrebungen, die sog. „Barrierefreiheit“, die vor allem aus der Architektur bekannt ist, auch im Internet zu etablieren, gibt es bereits seit den späten 90er Jahren des vergangenen Jahrhunderts. Aus dem Bereich der „Usability“ kommend, bezeichnet Barrierefreiheit die Anstrengungen, allen Menschen vollständigen Zugang auf Webseiten zu garantieren, unabhängig davon, wie die körperliche (oder mentale Verfassung) des Besuchers ist. Für sehbehinderte Menschen besteht neben der Verwendung der oben erwähnten Screenreader und Braille-Zeilen die Möglichkeit, Bildschirmlupen einzusetzen, die den angezeigten Inhalt eines Bildschirms stark vergrößern. Die meisten Betriebssysteme haben eine solche Funktionalität bereits integriert, es gibt jedoch eine Reihe von Drittanbieter-Software, die den Funktionsumfang des Lupen-Programms erweitern[5]. Zusätzlich verwenden sehbehinderte Menschen oft ein kontrastreiches Farbschema, um Inhalte besser aufzunehmen. Wichtig ist dabei, dass der Kontrast zwischen Vorder- und Hintergrundfarbe möglichst groß ist, damit Texte sich klar vom Rest der Webseite abheben. Verstärken lässt sich dieser Effekt auch durch die Invertierung (Umkehrung) der auf dem Bildschirm angezeigten Farben, sodass auch die bei einer Sehbehinderung oft auftretende Blendempfindlichkeit[6] nicht beim Lesen eines Textes stört.

Auch für motorisch eingeschränkte Menschen gibt es im Sinne der Barrierefreiheit Hilfsmittel, die es ihnen erlauben, gleichberechtigt im Internet zu agieren. So erlauben sog. Joystick-Mäuse die Bedienung eines Computers mittels eines Joysticks. Für körperlich noch stärker behinderte Menschen eignet sich eine sog. Mund-Maus. Mit einem schwachen Druck in eine Richtung wird die Maus bewegt, während Saugen und Blasen an der Maus einen Links, bzw. Rechtsklick auslöst. Die Eingabe von Text geschieht unter Verwendung einer Bildschirm-Tastatur.

Wie deutlich wird, existieren vielfältige Bemühungen, seh- und körperbehinderte Menschen am Internet teilhaben zu lassen. Wirft man jedoch einen Blick auf die Lage von gehörlosen Menschen, ist zu erkennen, dass für diese gesellschaftliche Gruppe kaum Anstrengungen unternommen werden, ihnen eine barrierefreie Teilhabe am World Wide Web zu ermöglichen[7]. Dass auch die Mehrheit der von Geburt an gehörlosen

5 Vgl. Moser et al., o. J.

6 Blendempfindlichkeit bezeichnet die Überstrahlung einer dunklen Schrift durch einen hellen Hintergrund, was dazu führt, dass er von betroffenen Personen nur schwer gelesen werden kann. Vgl. Moser et al., o. J.

7 Vgl. Klotz, 1998.

Menschen im Internet auf Hilfe angewiesen sind, ist ein wenig bekannter Fakt[8]. Denn „auch nach optimaler Förderung und Beschulung haben Gehörlose (…) meist einen eingeschränkten Wortschatz und erhebliche Unsicherheiten (…), was zu großen Schwierigkeiten und Missverständnissen beim Entschlüsseln komplexer schriftlicher Informationen führt.“[9]

Seit 2004 sind deshalb alle Online-Auftritte von Behörden der Bundesverwaltung verpflichtet, ihre Webseiten nach den Richtlinien der Barrierefreie-Informationstechnik-Verordnung (BITV) zu programmieren[10], die wiederum auf den Richtlinien der „Web Content Accessibility Guidelines“ (WCAG) beruht.[11] Neben den oben angeführten Punkten zur barrierefreien Programmierung von Webseiten schreibt die BITV Videos von gebärdenden Muttersprachlern vor, die sämtliche Texte einer Webseite übersetzen. Als Alternative besteht die Möglichkeit, Inhalte zusätzlich in der sog. „Leichten Sprache“ anzubieten. Dies ist eine Form „der schriftlichen [...] Kommunikation, die vor allem für [...] Menschen mit Lernschwierigkeiten entwickelt wurde. Bei Leichter Sprache geht es darum, dass Texte und Sprache einfach zu verstehen sind.“[12] Dies wird beispielsweise durch das Schreiben kurzer Sätze und den Verzicht auf Fremdwörter erreicht.

Auf den ersten Blick scheinen diese beiden Vorgehensweise ein funktionierendes Prinzip der Inklusion gehörloser Menschen zu sein. Schaut man jedoch genauer hin, fällt auf, dass diese Hilfestellungen nur in wenigen Fällen effektiv sind. Die Vorteile der Leichten Sprache (kurze, einfache Sätze ohne Nebensatz, keine Fremdwörter etc.) stellen gleichzeitig ihr größtes Defizit dar: Es ist nicht möglich, komplexe Sachverhalte, wie sie oft erläutert werden müssen, in der Leichten Sprache ohne Aussageverlust wiederzugeben. Für Fälle, bei denen diese Komplexität gefordert ist, bieten sich als Ausweg scheinbar Gebärden-Videos an.

Die Produktion solcher Videos ist jedoch ungleich aufwendiger und kostenintensiver, als die reine textbasierte Informationsvermittlung, da dafür neben der technischen Ausstattung und der erforderlichen Postproduktion auch ein(e) GebärdenmuttersprachlerIn

8 DGB, 2011, S. 3.

9 DGB, 2011, ebd.

10 Vgl. Bühler, 2005, S. 1.

11 Die WCAG sind eine Empfehlung zur Erstellung barrierefreier Benutzeroberfläche der Arbeitsgruppe „Web Accessibility Initiative“ (WAI) des „World Wide Web Consortiums“ (W3C). Das W3C wiederum ist ein Gremium zur Standardisierung von Webtechniken.
Die BITV regelt die Zugänglichkeit, also Barrierefreiheit eines Webauftritts; Webseiten, die sich an alle Regeln der BITV halten, gelten als barrierefrei.

12 Dworski, 2013, S. 1.

bezahlt werden muss[13]. Zusätzlich kommt hinzu, dass Videodateien bei gleichem Informationsgehalt um ein Vielfaches größer sind als Textdokumente und Server-Speicherplatz belegen, der anderweitig nicht mehr beansprucht werden kann. Ein weiterer Nachteil solcher Übersetzungs-Videos besteht in der Tatsache, dass Änderungen, wie sie an im Internet vorliegenden Texten oftmals durchgeführt werden, nicht auch im Video widergespiegelt werden können; ein einmal abgedrehtes Video kann nicht ohne erheblichen Aufwand geändert werden. Auch bei regelmäßigen und vielfältigen Veröffentlichungen, wie es bei News-Seiten der Fall ist, ist die Produktion von Gebärden-Videos allein schon aus zeitlichen und finanziellen Gründen kaum machbar. Dies führt dazu, dass vor allem auf Webseiten von Unternehmen und ganz besonders auf privaten Seiten in der absoluten Mehrzahl der Fälle auf Hilfsmittel für gehörlose Menschen verzichtet wird und diese dadurch – sofern sie nicht der schriftlichen Lautsprache mächtig sind – ausgeschlossen werden.

Experimente mit computeranimierten gebärdenden Avataren, die Anfang der 2000er Jahre unter anderem am Institut für Deutsche Gebärdensprache und Kommunikation Gehörloser der Universität Hamburg durchgeführt wurden, zielten darauf ab, die Erzeugung von Gebärden-Videos überflüssig zu machen. Ziel war, einen Avatar zu schaffen, der automatisiert vorgegebenen Text in Gebärdensprache darstellen können sollte[14]. Letztlich scheiterte das Projekt aber aus mehreren Gründen. Vor allem die abgehackten Bewegungen, die sich aus der mangelnder Rechenleistung ergab und die das von gehörlosen Menschen betriebene Lippenlesen unmöglich machten, sowie fehlende Emotionalität des Avatars und das Unvermögen, komplette Sätze sinnvoll zu übersetzen, führten dazu, dass Gehörlose die Avatare als nicht nützlich ablehnten.

Wird die Situation distanziert betrachtet, lässt sich erkennen, dass die Umsetzung des Barrierefreien Internets für gehörlose Menschen insbesondere für private Webseiten bislang als gescheitert eingestuft werden muss, sieht man von den oben erwähnten Übersetzungen in Leichte Sprache und Gebärden-Videos mit all ihren Nachteilen ab. Wie aber lässt sich die Inklusion gehörloser Menschen nun erreichen? Es ist an der Zeit, dass 25 Jahre nach dem Beginn des World Wide Web praktikable Lösungen gefunden werden, die es Gehörlosen ermöglichen, gleichberechtigt mit allen anderen Menschen am World Wide Web teilzunehmen.

13 Bühler, 2005, S. 120.

14 Vgl. auch Bühler, 2005, Seite 123.

1.2 Zielsetzung und Vorgehensweise

Wie im vorangegangenen Abschnitt deutlich gemacht wurde, besteht für gehörlose Menschen im Gegensatz zu blinden oder sehbehinderten Menschen nach wie vor keine Verfahrensweise zur einfach umsetzbaren Barrierefreiheit im World Wide Web. Angesichts der Tatsache, dass allein in Deutschland mindestens 80.000 gehörlose Menschen leben[15], ist es verwunderlich, dass bislang noch keine überzeugenden Übersetzungs-Technologien entwickelt wurden. Dienste wie „Google Translate“[16], die in den letzten Jahren kontinuierlich verbessert wurden, zeigen jedoch, dass es technisch möglich ist, komplexe Systeme wie (Laut-)Sprache von Maschinen interpretieren und übersetzen zu lassen. Für das Fehlen gleichwertiger Dienste für Gebärdensprache gibt es keine plausible Begründung, zumal diese Sprache im Vergleich zu Lautsprache sogar einfacher strukturiert ist. Dass dadurch der Ausschluss eines großen Bevölkerungsteils billigend in Kauf genommen wird, ist im 21. Jahrhundert nicht mehr akzeptabel.

Mit der vorliegenden Arbeit soll untersucht werden, ob der HTML5-Standard und die mit ihm neu eingeführten Elemente zur Inklusion (gehörloser) Menschen genutzt werden können und welche Möglichkeiten sich daraus ergeben. Dabei wird analysiert, wie sich Browser-Plugins wie Adobe Flash durch HTML5-Techniken gleichwertig ersetzen lassen, wobei das Augenmerk dabei auf barrierefreier Umsetzung liegt.

Für die vorliegende Literaturarbeit dienen unter anderem „Barrierefreies Webdesign“ von Christian Bühler (Herausgeber) und „Don't Make Me Think, Revisited“ von Steve Krug als Literaturgrundlage. Beide Werke bilden das Fundament für die Entwicklung gebrauchstauglicher und barrierefreier Internetauftritte, auch wenn „Barrierefreies Webdesign“, das bereits 2005 erschienen ist, in vielen Fällen technologisch überholt ist: Viele der indem Buch vorgestellten Empfehlungen lassen sich (dank HMTL5) besser und einfacher umsetzen.

Beginnend mit dem allgemeinen Begriff der „Usability“ und seinen verschiedenen Aspekten wird anschließend der Komplex „Barrierefreiheit im Webdesign“ beleuchtet und untersucht, wie sich Barrierefreiheit im Internet etabliert und welche Auswirkungen sie auf Webdesign hat. Darauffolgend wird analysiert, ob und wie der Standard „HTML5“ in der Lage ist, Barrierefreiheit besonders für gehörlose Menschen im Internet

---

15 Die Zahlen zu gehörlosen Menschen variieren stark, siehe http://www.eurosign.uni-hamburg.de/de/projektbeschreibung/2-beitrag/7-fakten-ueber-deutsche-gebaerdensprache-und-gehoerlose-in-deutschland.html

16 http://translate.google.com/

weiter zu verbessern. Speziell dem in HTML5 eingeführten „canvas“-Element wird besondere Beachtung geschenkt. Zur Demonstration soll als Nebenziel der Prototyp eines Übersetzungs-Programms für Webseiten konzipiert werden, das in der Lage ist, einzelne Wörter der Schriftsprache automatisch in statische Gebärden zu übertragen (Proof of Concept). Technologisch soll er neben dem „canvas“-Element auf weitere zeitgemäße Techniken wie HTML5-Tags und CSS3 aufbauen, um deren Eignung für moderne Projekte zu demonstrieren.

Die Arbeit richtet sich an Webentwickler und andere Personen, die über aktuelle Entwicklungen im Webdesign informiert sind. Aus diesem Grund wird auf die Definition spezifischer Fachtermini weitestgehend verzichtet, jedoch werden an einigen Stellen Begriffe (vornehmlich aus dem Bereich der Barrierefreiheit) für ein breiteres Publikum erklärt.

2 Barrierefreiheit mit HTML5

Jegliche Interaktion eines Anwenders mit einem Computer erfolgt über eine Mensch-Maschine-Schnittstelle, die als *user interface* (dt. Benutzeroberfläche) bezeichnet wird[17]. Sie besteht in der Regel aus einer grafischen Programmoberfläche, die dialoggesteuert arbeitet: Das Interface stellt dem Nutzer eine Reihe von möglichen Optionen in Form von Buttons, Fenstern, Menüs, Textbefehlen etc. vor, von denen er eine oder mehrere Optionen durch Klick mit der Maus/Auswahl mit der Tastatur oder durch einen Kommandozeilen-Befehl auswählt. Die Art und Weise, wie das user interface aufgebaut ist und wie schnell (effizient) der Anwender zu seinem gewünschten Ziel kommt, wird als *Usability* (dt. Benutzerfreundlichkeit/Bedienbarkeit, bzw. Gebrauchstauglichkeit) bezeichnet[18].

2.1 Usability

In der DIN EN ISO 9241 Teil 11 („Gestaltung von Bildschirmarbeit“) wird Usability unter Verwendung des deutschen Wortes „Gebrauchstauglichkeit“ wie folgt definiert:

> *„Gebrauchstauglichkeit ist das Ausmaß, in dem ein Produkt durch bestimmte Benutzer in einem bestimmten Nutzungskontext genutzt werden kann, um bestimmte Ziele effektiv, effizient und zufriedenstellend zu erreichen.“*

Nutzungskontext umfasst dabei „den Nutzer, seine Ziele […] [und] seine Ausrüstung am Arbeitsplatz“[19]. Oder, wie es Jakob Nielsen in seinem Werk „Designing Web Usability: The Practice of Simplicity“ sagt: „Usability ist der Grad an Qualität, in welchem der Benutzer die Interaktion mit etwas erlebt.[20]“ Ziel der Usability ist es also, dem Anwender die Abarbeitung seiner Aufgaben zufriedenstellend möglich zu machen, sodass er eine positive Einstellung gegenüber dem Produkt, mit dem er interagiert hat, bekommt. Dies führt als Konsequenz dazu, dass der Anwender das Produkt weiterhin

17 Bühler, 2005, S. 28.

18 Bühler, 2005, S. 46.

19 Bühler, 2005, ebd.

20 Nielsen, 1998, S. 12.

nutzen wird. Die Nichtbeachtung dieser Regel hat die Abwendung des Kunden zur Konsequenz: „Wenn der Kund ein Produkt nicht findet, wird er es nicht kaufen.“[21]

2.1.1 Frontend-bezogene Web-Usability

In seinem zum Standard etablierten Buch „Don't Make Me Think!“ fasst Steve Krugs den Begriff „Usability“ auf das Internet bezogen mit folgendem Satz zusammen:

> „A person of average […] ability and experience can figure out how to use the thing to accomplish something without it being more trouble than it's worth.“[22]

Usability soll also für den Standardbenutzer bei der Erfüllung einer Aufgabe nicht mehr Aufwand bedeuten, als unbedingt nötig ist.

Speziell auf das Internet und den direkten visuellen Output bezogen, wird die „Web-Usability“ deshalb in drei Bereiche unterteilt (die in dieser Form auch in den Richtlinien zur Barrierefreiheit[23] genannt werden):

Gestaltung der Webseite

Eine benutzerfreundlich gestaltete Seite muss *gut lesbar* sein. Das bedeutet, dass der Besucher in die Lage versetzt werden muss, alle Inhalte ohne horizontales Scrollen aufnehmen zu können. Dabei sollte auch auf die maximale Zeilenlänge von ca. 50 bis 70 Zeichen und die Verwendung von relativen Größenangaben sowie kontrastreichen Farben geachtet werden.

Gestaltung des Inhalts der Webseite

Empfehlungen zur Gestaltung einer Webseite zielen auf schnelle Lesbarkeit und Verständnis des Inhalts. Neben der Verwendung des umgekehrten Pyramiden-Prinzips, das besagt, dass die wichtigste Aussage am Anfang einer Seite stehen soll[24], sollte pro Abschnitt nur ein einzelner Gedanke formuliert werden, wobei ein einfacher und klar verständlicher Sprachstil gewählt werden sollte.

21 Nielsen, 1998, S. 8.

22 Krug, 2014, S. 9.

23 Vgl. Lynch, 2009.

24 Vgl. Bühler, 2005, S. 48.

Gestaltung des gesamten Webauftritts

Diese Forderung bezieht sich auf alle übergreifenden Elemente der Webseite und besagt, dass alle Unterseiten im gleichen Stil gehalten sein sollen, um dem Anwender Orientierung und Navigation zu vereinfachen.[25]

2.1.2 Technische Aspekte der Usability

Zu den eher inhalts- und gestaltungsorientierten Empfehlungen aus Kapitel 2.1.1 kommen eine Reihe von Richtlinien zum technischen Aufbau der Webseite:

Warte- und Ladezeit

Um ein Ziel „effektiv, effizient und zufriedenstellend zu erreichen[26]“, sollte eine Webseite innerhalb einer bestimmten Zeit geladen sein, damit der Anwender direkt seine Aufgabe bearbeiten kann. Lädt die Seite zu langsam, führt das zu negativen Emotionen gegenüber dem Produkt.

Schon 1968 beschrieb Robert B. Miller in seinem Paper „*Response time in man-computer conversational transactions*“ die Reaktionen von Anwendern auf unterschiedlich lange Ladezeiten von Programmen. Es lassen sich dabei drei Abschnitte von Ladezeiten trennen[27]:

- bis 0,1 Sekunden
  Alle Interaktion innerhalb von 0,1 Sekunden nimmt der Anwender als sofortige Reaktion wahr: „[…] the system is reacting instantaneously [...]“[28].
- bis 1,0 Sekunden
  Die Gedanken des Anwenders werden durch die Wartezeit gerade noch nicht abgelenkt. Nichtsdestotrotz bemerkt er die Verzögerung deutlich. „[...] [T]he user does lose the feeling of operating directly on the data.“[29]
- bis zehn Sekunden
  Zehn Sekunden sind das Limit, innerhalb dessen die Aufmerksamkeit des Nutzers auf dem aktuell angezeigten Dialog(-fenster) bleibt. Ladezeiten, die über

25 Vgl. Bühler 2005, S. 47 f.

26 DIN EN ISO 9241-11

27 Miller, 1968, S. 5.

28 Nielsen, 1993, S. 87.

29 Nielsen, 1993, ebd.

diese Wartezeit hinausgehen, müssen mit einer Anzeige versehen werden, die die zu erwartende Wartezeit ausgibt.

Ladezeiten webbasierter Applikationen und komplexer Seiten mit umfangreichen Tasks lassen sich in die gleichen Abschnitte einteilen. Anders die Dauer, die eine Webseite beim ersten Aufruf zum vollständigen Laden braucht: Maximal vier Sekunden ist im Jahr 2006 ein Besucher bereit, auf das Rendern einer (Shopping-)Webseite zu warten, bevor er sie unverrichteter Dinge verlässt (und sich eventuell der Konkurrenz zuwendet)[30]. Beachtenswert ist, dass dieser Wert sich im Vergleich zu früheren Untersuchungen zu Beginn des globalen Internetzeitalters um mehr als die Hälfte verringert hat[31]. Auf stark besuchten Seiten wie die von Google, Amazon oder Mozilla sind die Größen von Ladezeitwerten, die sich auf die Nutzerinter- und -reaktion auswirken, noch weitaus geringer und die Reaktionen sehr viel gravierender. So verzeichnet Amazon bei Erhöhung der Ladezeit um je 100 Millisekunden (0,1 Sekunden) einen Rückgang der Umsätze um je 1%[32], Google bei Erhöhung um 500 Millisekunden einen Besucherrückgang um 20%[33] und Mozilla eine Zunahme der weltweiten jährlichen Downloads von Firefox um 60 Millionen bei einer Reduzierung der Ladedauer der Download-Seite um 2,2 Sekunden[34].

Verwendung von Responsive Webdesign

Als Erweiterung der unter 2.1.1 genannten Empfehlung, Webseiten für alle Nutzer ohne Scrollen anzubieten, hat seit der Verbreitung von Smartphones die Verwendung von Responsive Webdesign große Bedeutung erlangt. Es ermöglicht, dass Webseiten reaktionsfähig (*responsive*) auf alle Bildschirmgrößen reagieren und somit bei optimaler Planung von allen Endgeräten aus gut bedient werden können. Erreicht wird dies durch den Verzicht pixelbasiert fester Objekt-Breiten und die Verwendung relativer Einheiten wie Prozent, „(r)em“, „vw“, „vh“ und „vmin“/„vmax“ für Größenangaben von Elementen und Schriften. Im Gegensatz zum Einsatz von prozentualen oder in „(r)em“-definierten Werten sind Einheiten, die sich auf die Breite („vw“) oder Höhe („vh“) (bzw. sich auf die sich daraus ergebenden kleinsten/größten Werte („vmin“/„vmax“)) des Browserfensters beziehen, dabei noch weit weniger verbreitet. Das liegt in erster Linie

30 Vgl. Akamai, 2006, S. 5.

31 Akamai, 2006, ebd.

32 Linden, 2006, S. 15.

33 Linden, 2006, ebd.

34 Cutler, 2010.

an der Unterstützung durch die gängigen Desktop-Browser, die erst seit Anfang 2013 durchgehend besteht[35].

Um Webseiten reaktionsfähig zu gestalten, ist der Einsatz von „Media Queries“ empfehlenswert. Diese erlauben es, CSS-Befehle nur für bestimme Bildschirmgrößen anwenden zu lassen[36]. So lassen sich gezielt Elemente auf kleinen Bildschirmen verkleinern, verschieben oder ganz ausblenden, um die Usability zu gewährleisten.

2.1.3 Usability und Barrierefreiheit

Bei Einhaltung sämtlicher (aber nur dieser) Web-Usability-Richtlinien wird eine Webseite zwar für sehende (hörende) und motorisch nicht eingeschränkte Menschen nutzbar sein und die in der DIN EN ISO 9421 definierten Vorgaben zur Nutzbarkeit erreichen, jedoch für Anwender mit körperlichen und/oder kognitiven Einschränkungen nicht derart barrierefrei sein, dass sie ohne Einschränkungen benutzt werden kann. Dies erklärt sich dadurch, dass viele wichtige Punkte der Barrierefreiheit durch Usability-Richtlinien erfüllt werden, spezielle Anforderungen wie der Einsatz von Alternativtexten für Grafiken und die Auszeichnung von Sprachwechseln aber nicht berücksichtigt werden. Dabei schränken umgesetzte Empfehlungen zur Barrierefreiheit die Usability nicht ein, sondern verbessern sie im Gegenteil weiter, da die Webseite für mehr gesellschaftliche Gruppen zugänglich und „gebrauchstauglich“ wird[37]. Im folgenden Kapitel wird untersucht, mit welchen technischen Möglichkeiten Barrierefreiheit hergestellt, bzw. verbessert werden kann.

2.2 Barrierefreiheit im Internet

Mit Einführung neuer Internettechnologien wie HTML5 und CSS3 ist es nun erstmals möglich, veraltete Technologien wie beispielsweise Flash- und Java-Applikationen zeitgemäß zu ersetzen. Neben dem Vorteil, nicht auf Plugins zur vollständigen Darstellung von Webseiten angewiesen zu sein, hat die Umstellung auf die sog. „nativen“ Techniken auch für körperlich eingeschränkte Menschen enorme Vorteile, wie sie bereits in der Einführung angedeutet wurden. Neben der Möglichkeit von CSS3, Animationen und Bewegungen ohne Einsatz von Flash oder JavaScript wiedergeben zu können – was dazu führt, dass die animierten Inhalte nun auch von Screenreadern gelesen wer-

35 Firefox 19 , Chrome 26, beide veröffentlicht Februar 2013, Sarafi 6.1 veröffentlicht Juni 2013, Internet Explorer 11 veröffentlicht Juni 2013 (Windowa 8.1), November 2013 (Windows 7).

36 Marcotte, 2011, S. 74ff.

37 Vgl. Bühler, 2005, 2005, S. 49

den können – ist insbesondere HTML5 geeignet, derzeit noch bestehende Barrieren weiter abzubauen.

Der Begriff „Barrierefreiheit“ ist dabei unlösbar mit dem Begriff der „Behinderung“ verbunden. Moderne Definitionen des Begriffs „Behinderung“ gehen sogar soweit, sie nur als Resultat einer nicht verfügbaren Barrierefreiheit zu sehen: „Je ungünstiger die Umweltbedingungen sind, desto eher erhält eine Beeinträchtigung das Gewicht der Behinderung.“[38] Es existieren eine Vielzahl verschiedener Behinderungen, die für betroffene Personen unterschiedlichste Auswirkungen haben können. In der folgenden Tabelle sind deshalb nur die Behinderungen aufgeführt, zu deren Verminderung an einem barrierefreien Internet gearbeitet wird. In der ersten Spalte wird die entsprechende Behinderung aufgeführt, die zweite und dritte Spalte enthalten Informationen über die mit der Behinderung verbundenen Einschränkungen und eine Liste von Hilfsmitteln, die ein betroffener Mensch zur Arbeit am Computer verwenden kann. Die letzte Spalte „Barrierefreiheit“ enthält Informationen darüber, welche technischen Möglichkeiten einem Webseiten-Betreiber gegeben sind; dabei verdeutlicht die Hintergrundfarbe der einzelnen Zellen den Aufwand, der zur Umsetzung der einzelnen Punkte nötig ist.

Behinderung		**Einschränkungen**	**Hilfsmittel**	**Barrierefreiheit**
Leichte geistige/kognitive Behinderung[39]		Von kaum bis leichte Intelligenzminderung. Schwierigkeiten, komplexe Sachverhalte aufzufassen durch vermindertes Abstraktionsvermögen	Hilfsperson bei Arbeit am Computer	Leichte Sprache
Körperliche Behinderung	Sehbehinderung/Blindheit	Mittleres bis absolutes Unvermögen, Inhalte visuell am Bildschirm zu erkennen/zu lesen	Bei Sehrest: Bildschirmlupen, kontrastreiches Farbschema, Screenreader Bei Blindheit: Screenreader, Braille-Zeile	„alt“-Attribut bei Grafiken, HTML-Elemente
				Untertitel für Videos
	Motorische Einschränkungen (der Hände)	Keine Möglichkeit, Computer über klassische Eingabemöglichkeit (Maus, Tastatur) zu steuern	Bildschirm-Tastatur, Mund-Maus, Joystick-Maus, Sprachsteuerung	Längere Timeouts, Möglichkeit der reinen Tastatur- oder Maus-Steuerung
	Schwerhörigkeit/Taubheit	Keine Verwendung der Lautsprache, Lese- und Schreibschwächen; dadurch vielfältige Einschränkungen im Alltag.	Bei Gehörlosigkeit: Gebärdensprache, *keine techn. Hilfsmittel!*	Leichte Sprache, Gebärden-Videos, Untertitel

Tabelle 1: Behinderungen mit den dabei auftretenden Einschränkungen und entsprechenden Mitteln zur Barrierefreiheit.

38 Bach, 1975.

39 Mittlere bis schwere geistige Behinderungen sind so schwerwiegend, dass die Benutzung des Internets für den Betroffenen nicht in Frage kommt.

Wie ersichtlich ist, differiert der Aufwand zur Erbringung der einzelnen Anforderungen stark. Während für sehbehinderte oder blinde Menschen die Angabe von Bildinhalten im „alt“-Attribut und die Verwendung von HTML5-Elementen[40] ohne großen Aufwand realisiert werden kann, ist die Erstellung von Untertiteln und die Implementierung einer zuverlässigen reine Steuerung per Tastatur/Maus schon mit größeren finanziellen und zeitlichen Aufwendungen verbunden. Als am umfangreichsten ist jedoch die Bereitstellung von Gebärden-Videos zu bewerten: Allein finanziell kommen auf Webseiten-Betreiber, die bereit sind, Gebärden-Videos anzubieten, hohe Kosten zu: Für ein professionell erstelltes Gebärden-Video fallen durchschnittlich 3500 € an, was bei 10 Minuten Länge einem Minutenpreis von ca. 350 € entspricht. Das Abfilmen eines Gebärdendolmetschers ist verhältnismäßig preiswerter, wird aber auch mit 500 € pro Video veranschlagt[41].

2.2.1 Barrierefreiheit im Webdesign mit HTML5

Die im letzten Kapitel in Tabelle 1 eingeführten Möglichkeiten zur barrierefreien Umsetzung von Webseiten sollen in diesem Kapitel näher beschrieben werden. Insbesondere der fünften Version der Auszeichnungssprache HTML (HTML5) wird besondere Beachtung geschenkt, da viele mit dieser Version eingeführten neuen Elemente das Potential haben, aufwendige bisher angewandte Verfahrensweisen abzulösen.

Um es Hilfsmitteln wie Braille-Zeilen und Screenreadern zu ermöglichen, möglichst alle Inhalte einer Webseite korrekt ausgeben zu können, existieren eine Reihe von grundlegenden strukturellen Empfehlungen beim Erstellen einer Webseite. Die wichtigsten sind im Folgenden kurz aufgeführt[42]:

Verwendung von HTML-Überschriften

Überschriften sind nicht nur durch ihre Gestaltung vom Rest des Textes abzuheben, sondern schematisch durch die Verwendung von `h1`-`h6`-Tags besonders zu kennzeichnen.

```
<span>Überschrift</span>
<p>Lorem ipsum</p>
```

40 Die Möglichkeiten, die sich aus der Verwendung von HTML5-Elementen ergeben, sind unter 2.2.1 erläutert.

41 BMAS, 2011, S. 5.

42 Alle hier aufgeführten Empfehlungen sind nicht nur zur Erstellung barrierefreier Webseiten gedacht, sondern wirken sich gleichermaßen positiv auf die Suchmaschinenoptimierung aus.

```
<style>
 span {
      font-size: 2em;
      color: #F77F00;
 }
 p {
      font-size: 1em;
      color: #000;
 }
</style>
```

Das Code-Beispiel erzeugt zwar eine visuell deutlich hervorgehobene Überschrift; für Screenreader und Suchmaschinen, die beide kein CSS interpretieren, hebt sich das Wort „Überschrift“ jedoch nicht vom Fließtext ab. Korrekt ist deshalb folgende Strukturierung:

```
<h1>Überschrift</h1>
<p>Lorem ipsum</p>
<style>
 h1 {
      font-size: 2em;
      color: #F77F00;
 }
 p {
      font-size: 1em;
      color: #000;
 }
</style>
```

Bei gleichbleibendem Aufwand ermöglicht dieses Code-Beispiel Screenreadern die korrekte Präsentation der Überschrift, während Suchmaschinen ebenfalls in die Lage versetzt werden, die Überschrift als solche wahrzunehmen und die Seite danach zu bewerten.

Verwendung korrekter, passender HTML(5)-Tags

Jeder Bereich einer Seite ist mit dem entsprechenden HTML(5)-Element zu wrappen, das am ehesten dafür geeignet ist. Die bedeutendsten Elemente sind dabei:

- `head` beinhaltet alle Meta-Angaben des Dokuments.
  Für Barrierefreiheit ist insbesondere die Vergabe eines seitenspezifischen „`title`“-Tag wichtig (`<title>Das ist der individuelle Titel</title>`).

- `body` beinhaltet den Hauptinhalt der Webseite.
  Dabei sollte mit dem „lang“-Attribut die verwendete Sprache definiert werden, damit Screenreader den Inhalt korrekt vorlesen. Hinweis: Mit HMTL5 kann jedes Element einer Webseite eine spezifische Sprache zugewiesen bekommen (`<body lang="de">...</body>`).

- `header` beinhaltet den Kopfteil eines Dokuments (Logo, Titel, Meta-Angaben über den Autor, Veröffentlichungs-Datum, zugewiesene Kategorie etc.).

- `main` definiert den Hauptinhalt der Webseite.
  Vielfach gefordert werden unsichtbare Anker-Links im Kopfbereich einer Webseite, die den Nutzer bei Klick die Navigation überspringen lassen und ihn direkt zum eigentlichen Inhalt führen. Mit zunehmender Verbesserung der Screenreader und bei korrekter Verwendung des `main`-Tags werden diese „Skip to Content“-Links überflüssig, da der Screenreader auf Aufforderung selbstständig zum Beginn des `main`-Elements springen kann.

- `nav` beinhaltet ausschließlich Navigations-Links:

```
<nav>
        <ul>
             <li><a href="seite_1.php" title="Seite 1">Seite 1</a></li>
             <li><a href="seite_2.php" title="Seite 2">Seite 2</a></li>
             <li><a href="seite_3.php" title="Seite 3">Seite 3</a></li>
       </ul>
</nav>
```

  Die Verwendung von <nav> stellt sicher, dass Screenreader den Bereich zuverlässig als Navigations-Block erkennen und bei Bedarf überspringen können.

- `section` beschreibt einen Abschnitt des Dokuments.

- `article` beschreibt einen Abschnitt des Dokuments, *der unabhängig von den übrigen Inhalten existieren kann.*
- `aside` beinhaltet weniger wichtigen Content, ohne den der eigentliche Inhalt trotzdem verständlich bleibt. Er enthält beispielsweise Glossare (innerhalb eines Artikels) oder Links zu ähnlichen Beiträgen (außerhalb eines Artikels).
- `footer` definiert den Fußteil einer Webseite mit Links zu Impressum etc. Meist enthält der Footer weniger wichtige Links, die ein Screenreader bei Verwendung des `footer`-Element richtig behandeln kann (z. B. nur auf Verlangen vorlesen). Innerhalb eines Artikels verwendet, kann das `footer`-Element ähnlich wie `header` auch Meta-Angaben beinhalten.

Texthinterlegungen für Grafiken

In Anforderung 1 der BITV wird vorgeschrieben, dass für „jeden Nicht-Text-Inhalt, der dem Nutzer oder der Nutzerin präsentiert wird, [...] eine Text-Alternative […] [bereitgestellt wird], die den Zweck dieses Inhalts erfüllt.“[43] Jedes in einer Seite eingebundene Bild soll also über ein den Inhalt äquivalent beschreibendes `alt`-Attribut verfügen, wobei diese nicht länger als 150 Zeichen sein sollte[44]. Sehbehinderte und blinde Menschen werden durch das Setzen des `alt`-Attributs in die Lage versetzt, durch einen Screenreader die dargestellte Information vermittelt zu bekommen:

```
<img src="hund.jpg" width="150" height="100" alt="Ein
schwarzer Hund auf grüner Wiese" />
```

Durch das mit HTML5 eingeführte `figure`-Element haben Webentwickler eine weitere Möglichkeit, Grafiken mit Text-Alternativen zu versehen, bekommen:

```
<figure>
 <img src="hund.jpg" width="150" height="100" alt="Ein
schwarzer Hund auf grüner Wiese" />
 <figcaption>Ein schwarzer Hund sitzt auf einer grünen
```

43 BITV 2.0, Anforderung 1.1.1

44 Vgl. Bühle 2004, S. 55.

```
Wiese und wedelt mit dem Schwanz.</figcaption>
</figure>
```

Zu beachten ist hierbei das `figcaption`-Element, dass innerhalb von `figure` verwendet werden kann, um Grafiken (ausführlich) zu beschreiben. Text innerhalb diesen Elements ist für alle Anwender sichtbar, weshalb er auch dafür genutzt werden kann, ergänzende Informationen zu vermitteln, die in der eigentlichen Grafik nicht sichtbar sind. Obwohl das `figcaption`-Element das `alt`-Attribut nicht überflüssig macht, können Screenreader auf das Element ausweichen, wenn kein alternativer Text mit dem `alt`-Attribut gesetzt wurde.

Texthinterlegungen für multimediale Inhalte

In der BITV findet sich auf multimediale Inhalte bezogen die Anforderung, dass für jede Audio-Datei eine „Text-Alternative mit gleichwertigen Informationen“ und für jede Video-Datei eine „Text-Alternative *oder eine Tonspur* mit gleichwertigen Informationen“ bereitgestellt werden muss[45], ebenso wie Untertitel. Im Vergleich zu HTML 4.01, wo Audio-Spuren und Videos nicht nativ in eine Webseite eingebunden werden konnten, sondern mithilfe von Adobe Flash, bzw. innerhalb eines iFrames oder mit dem `object`-Elements platziert wurden, können mit HTML5 (audio-)visuelle Inhalte nativ eingebunden werden. Dies vereinfacht auch die Handhabung der geforderten Untertitel erheblich, die mit dem „`track`“-Element gesetzt werden.

```
<video controls>
 <source src="foo.ogg" type="video/ogg">
 <source src="foo.mov" type="video/quicktime">
 <track kind="captions" src="sampleCaptions.vtt"
srclang="en">
 <track kind="descriptions" src="sampleDesciptions.vtt"
srclang="en">
 <track kind="chapters" src="sampleChapters.vtt"
srclang="en">
 <track kind="subtitles" src="sampleSubtitles_de.vtt"
srclang="de">
```

45 BITV 2.0 Anforderung 1.2.1

```
 <track kind="metadata" src="keyStage1.vtt"
srclang="en" label="Key Stage 1">
</video>
```

Das `track`-Element verfügt über die Attribute „`kind`“, „`src`“, „`srclang`“, „`label`“. Ersteres definiert die Art des Untertitels. Folgende Werte können verwendet werden:

- `subtitles` für Untertitel als Übersetzung anderer Sprachen
- `captions` enthalten Transkriptionen des Audio-Materials sowie Beschreibungen der Geräuschkulisse und Sound-Atmosphäre
- `descriptions` enthalten Beschreibung des Film-Materials
- `chapters` legen Zeitfenster fest, zu denen gezielt gesprungen werden kann
- `metadata` erlaubt JavaScript-Zugriff auf die Untertitel

Mit `src` wird der Pfad zu einer Text-Datei definiert, die die Untertitel enthält, welche im „WebVTT“-Format zur Verfügung gestellt werden sollten[46]. `scrlang` definiert die Sprache des `track`-Elements, während `label` den Titel des spezifischen Untertitels enthält, den der Browser anzeigt. Da sich das `track`-Element noch in der Entwicklung befindet und deshalb nicht weit verbreitet ist[47] (Juni 2014) und das WebVTT-Format bislang nur von einer Community-Arbeitsgruppe beschrieben und noch nicht in den W3C-Standardisierungs-Prozess aufgenommen wurde, soll WebVTT an dieser Stelle näher erläutert werden:

WebVTT-Dateien sind reine Textdateien, die mit dem String „WEBVTT“ begonnen werden, der von einer Leerzeile gefolgt werden muss:

```
WEBVTT

1 - Szene 1a
00:01:14.815 --> 00:01:18.114
<v Person A>- Lorem?
```

46 Vgl Pfeiffer 2014.

47 Chrome 23, Firefox 31, Internet Explorer 10, Opera 12.10, Safari 6.

```
<v Person B>- Lorem ipsum?

2 - Szene 1b
00:01:18.171 --> 00:01:20.991
<v Person C>- Lorem ipsum.

3 - Szene 2
00:01:21.058 --> 00:01:23.868
- [lautes Geräusch]
<v Person B>- Lorem ipsum dolor sit <c.red>amet</c>,
consectetur adipisic.
```

Die optional nummerierten und betitelten „Cues“ bestehen aus zwei Zeitangaben getrennt mit dem String „`-->`“, die die Start- und Endzeit eines Cue-Blockes definieren. Darunter befindet sich der anzuzeigende Text. Dieser kann mit einigen begrenzten CSS-Eigenschaften gestylt werden: Der `v`-Tag spezifiziert dabei eine „Person“, während der `c`-Tag den beinhalteten Text einer CSS-Klasse (in diesem Fall „`red`“) zuweist.

```
<style>
 video::cue {background-color: #fff; color: black;}
 video::cue(v[voice="Person B"]) {color: #F77F00;}
 video::cue .red {color: #F00;}
</style>
```

Aktuell (Juni 2014) ist die Browser-Unterstützung für das `cue`-Pseudo-Element noch sehr gering, da derzeit noch aktiv an dem Beschreibung gearbeitet wird, weshalb sich auch die Syntax sowohl des WebVTT-Formats als auch die des `cue`-Pseudo-Element unangekündigt ändern kann. Obwohl deshalb von einem Einsatz in einem Produktiv-System derzeit noch abzuraten ist, wird sich die native Video- und Audio-Einbindung unter Zuhilfenahme des `track`-Elements im Sinne der Barrierefreiheit durchsetzen.

3 Ansätze zur Barrierefreiheit für Gehörlose

Wie im vorangegangenen Kapitel deutlich wurde, differieren die Möglichkeiten stark, für die Gruppe der körperlich beeinträchtigten Menschen barrierefreie Webseiten anzubieten. Besonders gehörlosen Menschen, die lautsprachliche Schwierigkeiten haben, können Inhalte nur in Form von Leichter Sprache, bzw. Gebärden-Videos vermittelt werden mit allen oben angeführten Nachteilen. Im diesem Kapitel soll deshalb untersucht werden, wie das HTML5 „canvas“-Element geeignet ist, gehörlosen Menschen den Zugang zu online erhältlichen Informationen zu vereinfachen. Dafür wird das Element im folgenden Unterkapitel vorgestellt und die grundlegende Anwendung erläutert. Im darauffolgenden Kapitel wird die prinzipielle Funktionsweise eines automatisiert arbeitenden Übersetzungs-Programm basierend auf dem „canvas“-Element skizziert.

3.1 Das „canvas“-Element

Unter den vielen mit HTML5 verfügbaren Neuheiten – deren prominentestes Feature die oben angeführte native Wiedergabe von Audio- und Videodateien ist – zählt das „canvas“-Element zu den wichtigsten Neuerungen. Mithilfe diesen Elements lassen sich direkt im Browser praktisch uneingeschränkt dynamische/interaktive grafische Elemente ohne den Bedarf von Plugins erzeugen. Alle Grafiken werden dabei mittels JavaScript generiert und manipuliert, was es möglich macht, auf Eingaben des Nutzers zu reagieren. So lässt sich beispielsweise der Mauszeiger verfolgen und an seiner jeweiligen Position eine Markierung setzen – auf diese Weise entsteht ein virtueller Pinsel[48]. Aber auch umfangreichere Projekte wie ganze browserbasierte (Jump’n’Run)-Spiele[49] oder Online-Tools beispielsweise zur Erstellung von Favicons[50] lassen sich mit dem „canvas“-Element umsetzen.

Grundlegend wird der Einsatz des „canvas“-Element wie folgt im HTML-Code vorbereitet:

```
<canvas id="first_canvas" width="500" height="300"></canvas>
```

48 Ein simples Beispiel: http://tricedesigns.com/portfolio/sketch/brush.html

49 Als prominentes, „canvas“-basiertes Spiel sei hier „Pirates Love Daisies“ genannt (http://www.pirateslovedaisies.com/)

50 http://www.xiconeditor.com/

Das „canvas“-Element wird wie alle anderen HTML-Elemente mit den Vergleichszeichen `<` und `>` gekennzeichnet. Neben der Angabe einer ID (`id="first_canvas"`), um das Element mit JavaScript ansprechen zu können, sind noch die Werte für Breite und Höhe festgelegt (`width="500" height="300"`).

Rechteck

Um dieses erstellte Element tatsächlich mit Inhalten (einem simplen Rechteck) zu versehen, wird der folgende JavaScript-Code verwendet:

```
var canvas = document.getElementById("first_canvas");
var context = canvas.getContext("2d");
context.fillStyle = "#ffffaa";
context.fillRect(0, 0, 500, 300);
```

Die erste Zeile sucht im HTML-Dokument nach einem Element mit der ID „`first_canvas`“, die zweite Zeile ruft dann in diesem Element den 2D-Kontext auf, auf dem im „canvas“-Element alle Inhalte gezeichnet werden[51]. Auf diesem Kontext wird nun ein Rechteck gezeichnet. `fillStyle` setzt die Farbe als hexadezimalen Wert vor, `fillRect(0, 0, 500, 300)` zeichnet in der gesetzten Farbe in der linken oberen Ecke ein Rechteck der Größe 500px x 300px.

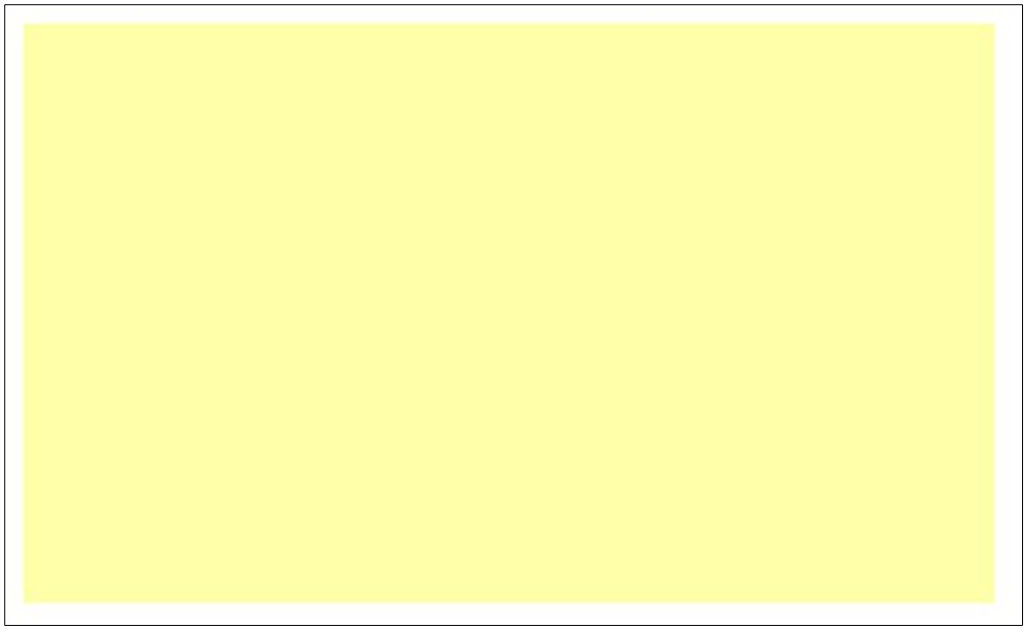

Abbildung 1: gezeichnetes Rechteck "first_canvas" (Screenshot)

51 Alternativ kann hier auch WebGL aktiviert werden, was die Erstellung und Manipulation von 3D-Objekten ermöglicht.

Pfade

Eine weitere wichtige Methode sind Pfade. Zum Zeichnen eines Pfades eignet sich folgender Code:

```
var canvas = document.getElementById('second_canvas');
var context = canvas.getContext('2d');
context.beginPath();
context.moveTo(50, 50);
context.lineTo(150,150);
context.lineTo(180,150);
context.lineWidth = 10;
context.strokeStyle = 'black';
context.stroke();
```

Die ersten beiden Zeilen sind identisch zum ersten Beispiel; sie werden ausnahmslos für jede Darstellung von „canvas“-Zeichnungen benötigt. Die dritte und vierte Zeile beinhalten die Anweisungen, dass ein mehrteiliger Pfad gezeichnet werden soll. `moveTo(50, 50)` setzt den Beginn des Pfades auf die Position 50, 50 (X- und Y-Werte) und legt somit gleichzeitig den Startpunkt des Pfades fest. Mit `lineTo(150, 150)` wird der Endpunkt des ersten Pfades auf der Position 150,150 festgesetzt. Da in diesem Beispiel keine Startpunkt-Definition für den zweiten Pfad angegeben ist, setzt er direkt am Endpunkt des ersten Pfades an und verläuft von dort bis zur Position 180, 150. Mit der Angabe `lineWidth` wird die Breite des zu zeichnenden Pfades festgelegt, mit `strokeStyle` die zu verwendenden Farbe und mit `stroke()` wird der Pfad schließlich gezeichnet.

Der Code führt zu dem unter Abbildung 2 eingebunden Ergebnis.

„canvas“-Zeichnungen können aus beliebig vielen aneinander ansetzenden Pfaden bestehen. Da sie jedoch keine Informationen über Kurvenverhalten kennen und deshalb nur geradlinig von Punkt A zu Punkt B verlaufen können, sind sie zur Darstellung komplexer, detaillierter Zeichnungen nur wenig geeignet.

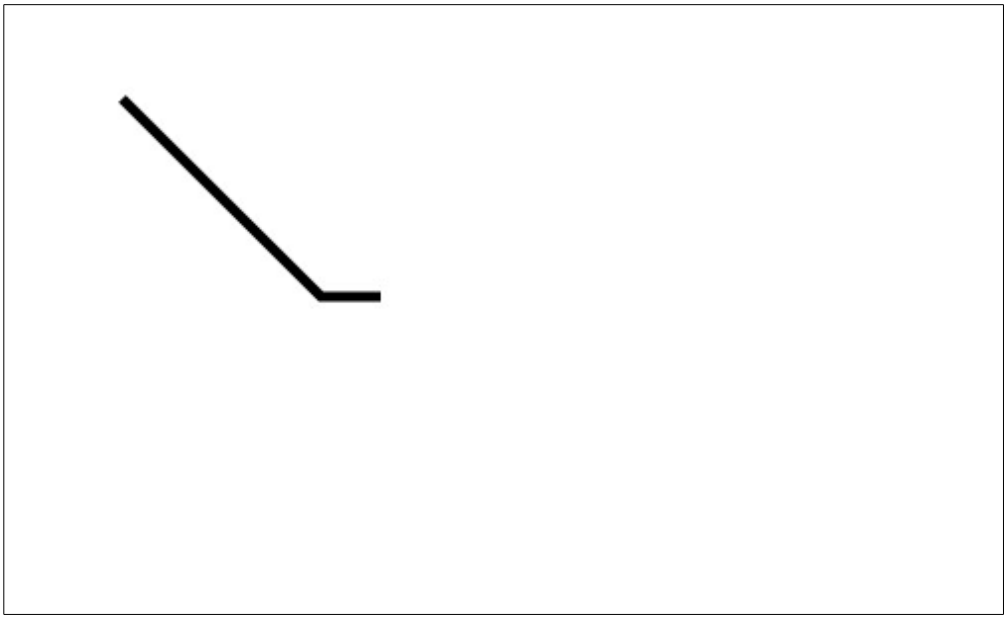

Abbildung 2: gezeichneter Pfad "second_canvas" (Screenshot)

Bézierkurven

Abhilfe schaffen sog. Bézierkurven. Diese bestehen aus einem Start- und Endpunkt und mindestens einem dazwischenliegenden Kontrollpunkt, zwischen dem/denen die Kurve „aufgespannt“ wird. Sie kann im „canvas“-Element mit folgendem JavaScript-Code erzeugt werden:

```
var canvas = document.getElementById('third_canvas');
var context = canvas.getContext('2d');
context.beginPath();
context.moveTo(188, 130);
context.bezierCurveTo(140, 10, 388, 10, 388, 170);
context.lineWidth = 10;
context.strokeStyle = 'black';
context.stroke();
```

Im Vergleich zum vorherigen Beispiel ist nur die fünfte Zeile anders, während alle anderen Angaben gleich bleiben. Die Angabe `bezierCurveTo(140, 10, 388, 10, 388, 170)` definiert mit dem ersten Zahlenpaar die Position des ersten Kontrollpunktes, mit dem zweiten Zahlenpaar die des zweiten Punktes und mit dem dritten Paar schließlich die des Endpunktes.

Der Code führt zu folgendem Ergebnis:

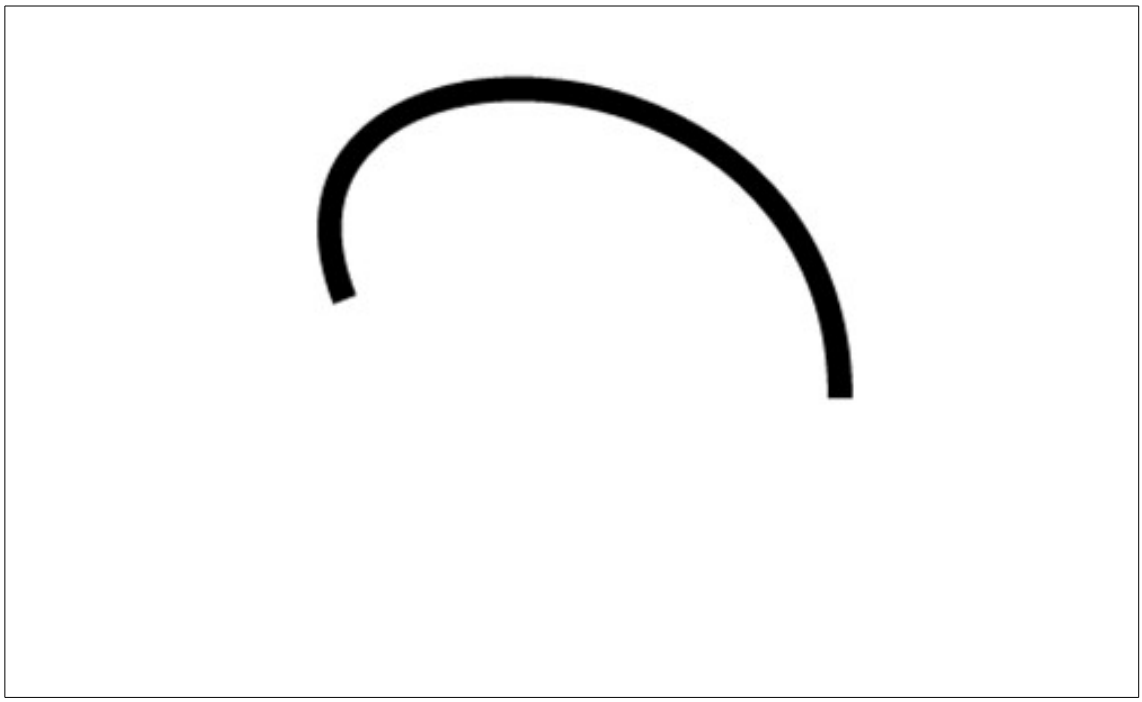

Abbildung 3: gezeichnete Bézierkurve "third_canvas" (Screenshot)

Im Gegensatz zu einzelnen Pfaden haben Bézierkurven den großen Vorteil, dass auch komplexe Formen und Kurven mit nur wenigen Positionsangaben gezeichnet werden können. Das bedeutet, dass die Anzahl der benötigten Wert-Zeilen drastisch abnimmt. Folgendes Beispiel verdeutlicht dies an der Abbildung einer vektorbasierten Hand:

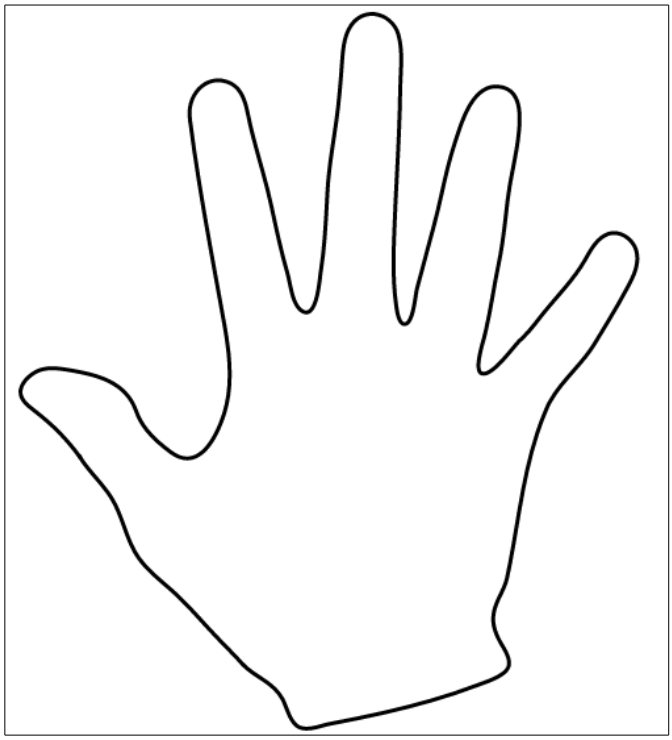

Abbildung 4: vektorbasierte Hand

Für die pfadbasierte Darstellung der oben abgebildeten Hand werden knapp 1210 Zeilen mit `lineTo`-Angaben (und 2420 X- und Y-Positionen) benötigt, während die Bézierkurven-basierte Version mit 60 Zeilen „`bezierCurveTo`“ auskommt, was einer absoluten Anzahl von 360 Positionsangaben entspricht. Neben der enormen Reduktion der benötigten Werte nimmt auch die Größe der entsprechenden Datei deutlich ab. Die nachfolgende Tabelle verdeutlicht diesen Unterschied und bezieht als zweiten Vergleich die Abbildung als Rastergrafik mit ein.

	pfadbasierte Grafik	**bézierkurvenbasierte Grafik**	**Rastergrafik**
Größe (x1)	~ 40 Kilobyte	~ 5 Kilobyte	~ 16 Kilobyte
Größe (x1.000)	~ 40 Megabyte	~ 5 Megabyte	~ 16 Megabyte
Größe (x18.000)	~ 720 Megabyte	~ 90 Megabyte	~ 288 Megabyte

Tabelle 2: Unterschiedliche Größe und Ladezeiten von Grafiken

Die drei Zeilen „Größe x1“, „Größe (x1.000)“ und „Größe (x18.000)“ enthalten die voraussichtlichen Größen der Datei mit Informationen zur Zeichnung der Hand bei einer, 1.000 und 18.000[52] gespeicherten Zeichnungen. Wie deutlich wird, führt die Verwendung von Bézierkurven in jedem Fall zum geringsten Speicherverbrauch. Die im Vergleich zur pfadbasierten Grafik deutlich geringere Größe der Rastergrafik erklärt sich dabei durch die simple Ausgangsform; bei komplexer werdenden Abbildungen nimmt die Größe rasch zu.

Weitere Objekte, die im „canvas“-Element gezeichnet werden können, sind Kreise, Pfeile, Rastergrafiken und Texte. Für den in dieser Arbeit untersuchten Einsatz zur Unterstützung gehörloser Menschen im Internet sind diese Formen jedoch irrelevant, da sich Bézierkurven am besten zur Darstellung freier Formen wie Gebärden eignen.

3.1.1 JavaScript-Bibliotheken zur vereinfachten Manipulation

Seit Beginn der Arbeit an HTML5 und dem „canvas“-Element entstanden eine Vielzahl von Ansätzen, die dazu dienen, die Anwendung des Elements zu vereinfachen. Sie alle bestehen aus JavaScript-Bibliotheken, die die Anzahl der vom Programmierer zu

52 Das große Wörterbuch der Deutschen Gebärdensprache, welches auf DVD erhältlich ist, enthält 18.000 Wörter in Gebärden-Videos und dient daher als Vergleich.

schreibenden Code-Zeilen reduzieren sollen: „Instead of operating on [...] low level, [the libraries] provide[s] simple but powerful object model on top of native methods, […] take[s] care of canvas state and rendering, and let[s] us work with “objects” directly.“[53] Die meisten JavaScript-Bibliotheken sind für die Arbeit mit interaktiven Animationen ausgelegt und ermöglichen die Erstellung komplexer „canvas“-basierter Oberflächen.

In einer vom Urheber der Bibliothek „fabric.js“[54] – Juriy Zaytsev – verwalteten Tabelle[55] sind die meisten Programme inklusive ihrer Eigenschaften eingetragen. Auch die Webseite „JSter“ bietet einen Überblick über die erhältlichen „canvas“-Bibliotheken[56]. Beide Tabellen ermöglichen einen schnellen Überblick über unter anderem die letzte Aktualisierung, die verwendete Lizenz und die jeweilige Größe der Bibliothek. Aus der Distanz des letzten Veröffentlichung-Datums einer Version zum aktuellen Datum lässt sich dabei erkennen, wie aktiv die Bibliothek gepflegt wird, was generell von Vorteil ist, weil angenommen werden kann, dass die entsprechende Bibliothek auch in Zukunft aktiv weitergepflegt wird. Die wichtigsten sind in folgender Tabelle zusammengefasst:

Name der Bibliothek	**Version (Datum)▾**	**Größe (Kilobyte)**	**Geeignet für**	**SVG-Parser**
pixi.js	1.5.3 (24.04.2014)	87	Animationen	Nein
Fabric.js	1.4.5 (14.04.2014)	77 – 225	Komplette Bereich	Ja
Paper.js	0.9.18 (07.04.2014)	167 - 200	Animationen	Ja
Kinetic.js	5.1.0 (27.03.2014)	109	Wie Fabric.js, komplexer	Teilweise
Easel.js	0.7.1 (12.12.2013)	79	Animationen	Nein
Two.js	0.3.0 (25.10.2013)	50	Animationen	Ja

Tabelle 3: Übersicht ausgewählter JavaScript-Bibiotheken zur Manipulation des „canvas“-Elements.

Im folgenden Abschnitt soll untersucht werden, wie sich die ausgewählten Bibliotheken für den Übersetzungs-Prototypen eignen.

Anhand der Spalte „**Geeignet für**“ in oben angeführter Tabelle 3 wird ersichtlich, dass für den Prototypen der Einsatz einer JavaScript-Bibliothek, die vornehmlich für Animationen entwickelt wurde, nicht geeignet ist, da die Animierung des „canvas“-Elements in

53 Zaytsev, 2012.

54 http://fabricjs.com/

55 https://docs.google.com/spreadsheet/ccc?key=0Aqj_mVmuz3Y8dHNhUVFDYlRaaXlyX0xYSTVnalV5ZlE#gid=0

56 http://jster.net/category/canvas-wrappers

dieser Arbeit nicht vorgesehen ist. Im Folgenden soll deshalb besonderer Augenmerk auf die Bibliotheken „Fabric.js“ und „Kinetic.js“ gelegt werden.

Fabric.js

Um das gleiche visuelle Ergebnis wie unter Abbildung 2 abgebildete Pfad zu erhalten, wird mit Fabric.js folgender Code verwendet:

```
var canvas = new fabric.StaticCanvas('fabric_canvas');
var line = new fabric.Line([50, 50, 150, 150]);
line.set ({
 stroke: 'black',
 strokeWidth: 5,
});
var line2 = new fabric.Line([150, 150, 180, 150]);
line2.set ({
 stroke: 'black',
 strokeWidth: 5,
});
canvas.add(line);
canvas.add(line2);
```

In der ersten Linie wird definiert, welche ID das „canvas“-Element hat, und dass Fabric.js es ansprechen soll. Dabei ist wichtig, dass ein Objekt, das über keine Interaktivität verfügen soll, mit `fabric.StaticCanvas()` statt `fabric.Canvas()` initialisiert werden sollte, weil dies eine einfachere Version des „canvas“-Elements erzeugt. Zaytsev schreibt dazu: „This creates a “lighter” version of canvas, without any event handling logic.“[57]

Mit `var line = new fabric.Line(`[…]`)` wird eine einteilige Linie begonnen, die zu den in den eckigen Klammern festgelegten Koordinaten läuft. In der dritten Zeile wird ein Array geöffnet, dass weitere Angaben zum Pfad enthält (Farbe und Dicke). Für den zweiten Abschnitt der Linie wiederholt sich der Code auf den Zeilen 7 – 11. Mit `canvas.add(line[2])` werden die beiden Linien schließlich gerendert.

57 Zaytsev, 2012.

Wie ersichtlich ist, benötigt Fabric.js für jedes Pfadsegment ein eigenes Objekt (`line1` und `line2`), die mit einem eigenen Array definiert werden müssen. Bei umfangreicheren Grafiken, die aus dutzenden Ankerpunkten bestehen, erhöhen sich dadurch die benötigten Code-Zeilen drastisch.

Zur Generierung von Bézierkurven wird das Objekt `fabric.Path` in Verbindung mit einer Reihe von Koordinaten verwendet. Dieses Objekt (path dt. Pfad) ist von Zaytsev eng an den SVG-Standard[58] angelehnt, wodurch es mit dem gleichen Syntax wie das SVG Path-Element benutzt werden kann[59]. Die unter Abbildung 3 gezeichnete Bézierkurve lässt sich mit folgendem Code unter Fabric.js darstellen:

```
var canvas = new fabric.StaticCanvas('fabric_canvas');
var bezier = new fabric.Path('M 188 130 C 140, 10, 388,
10, 388, 170');
bezier.set ({
 stroke: 'black',
 strokeWidth: 5,
});
canvas.add(bezier);
```

Der einzige Unterschied zum vorangegangenen Beispiel liegt im Aufruf des fabric.Path-Objekts, das mit einer Reihe von Koordinaten versehen ist:

- M steht für Move und setzt den Beginn des Pfades auf den Wert 188, 130
- mit C beginnt eine Bézierkurve, die genau wie im dritten „canvas“-Beispiel drei Koordinaten-Paare erwartet.

Die weiteren Zeilen sind identisch zum darüber gezeigten Beispiel. Fabric.js stellt die gezeichnete Bézierkurve jedoch nicht als Pfad dar, sondern schließt die Kurve vom letzten zum ersten Punkt, sodass eine gefüllte Fläche gebildet wird.

Eine simple Animation eines Rechtecks wird mit folgendem Code umgesetzt:

58 Definition SVG (in Einleitung?)

59 Vgl. Zaytsev, 2012.

```
var canvas = new fabric.StaticCanvas('fabric_canvas');
var rect = new fabric.Rect({
 left: 0,
 top: 100,
 fill: 'red',
 width: 20,
 height: 20
});
rect.animate('left', 470, {
 onChange: canvas.renderAll.bind(canvas),
 duration: 2000,
 easing: fabric.util.ease.easeOutExpo
});
canvas.add(rect);
```

Das in der zweiten Zeile initiierte `fabric.Rect`-Objekt (Rect dt. Rechteck) wird mit einem Array von Angaben über Position, Farbe und Größe definiert. Die darauffolgende Zeile ruft die in Fabric.js integrierte Animations-Methode auf, die das Rechteck auf den festgelegten Wert „470 Pixel“ verschiebt. Das dritte Argument ist optional und spezifiziert genauere Details der Animation.[60] In dem vorliegenden Fall wird mit `canvas.renderAll` bei jedem Frame das „canvas“-Element neu gezeichnet, die Dauer der Animation auf 2000 Millisekunden (2 Sekunden) gesetzt und der Verlauf der Animation (`easing`) definiert (`easeOutExpo`). Die letzte Zeile ist identisch zum vorhergehenden Beispiel und zeichnet das Rechteck auf das „canvas“-Element.

Die Ausführung des Codes führt zu folgendem visuellen Ergebnis:

60 Vgl. Zaytsev, 2012.

Abbildung 5: Das Rechteck bewegt sich innerhalb von zwei Sekunden von links nach rechts.

Fabric.js eignet sich auch, um externe Bilder im „canvas“-Element darzustellen. Die Grafiken können dann wie alle anderen Objekte manipuliert und animiert werden. Folgender Code lädt ein Bild aus dem DOM und stellt es als Objekt im „canvas“-Element dar:

```
var canvas = new fabric.StaticCanvas('fabric_canvas');
var imgElement = document.getElementById('image');
var image = new fabric.Image(imgElement);
canvas.add(image);
```

Dazugehöriger HTML-Code:

```
<img src="hund.jpg" id="image">
```

Neben der schon beschriebenen `canvas`-Variablen wird mit der zweiten Variable die Methode `getElementById()` aufgerufen, die im DOM nach der angegebenen ID `image` sucht und sie übernimmt. In der dritten Zeile wird das Fabric.js-Image-Objekt (Image dt. Bild) mit dem Wert der zweiten Variable aufgerufen und in der vierten Zeile auf das „canvas“-Element gezeichnet. Daraus resultiert folgendes Ergebnis:

Abbildung 6: Auf das „canvas“-Element gerenderte Grafik.

Kinetic.js

Obwohl der Name der Bibliothek anderes vermuten lässt (kinetic dt.: „Bewegung…“), eignet sich das Tool auch zur vereinfachten Anwendung von statischen „canvas“-Elementen. Abbildung 2 wird mit Kinetic.js wie folgt umgesetzt:

```
var stage = new Kinetic.Stage({
 container: 'kinetic_canvas',
 width: 500,
 height: 300
});
var layer = new Kinetic.Layer();
var line = new Kinetic.Line({
 points: [50, 50, 150, 150, 180, 150],
 stroke: 'black',
 strokeWidth: 5,
 lineJoin: 'round'
});
layer.add(line);
stage.add(layer);
```

Dazugehöriger HTML-Code:

```
<div id="kinetic_canvas"></div>
```

Im Vergleich zu den vorherigen Beispielen von Fabric.js wird deutlich, dass Kinetic.js eine Ebene höher ansetzt und selbstständig das erforderliche „canvas“-Element in das DOM-Baum[61] schreibt. Um zu wissen, in welchen Bereich der Webseite das Element gesetzt werden soll, benötigt die Bibliothek ein `div`-Element mit der im JavaScript referenzierten ID. In den ersten fünf Zeilen wird die erforderliche Bühne (engl. stage) mit der ID definiert und mit Breiten- und Höhenangaben versehen. `var layer = new Kinetic.Layer()` definiert eine Ebene, auf der die einzelnen Objekte (in diesem Fall nur eine Linie) liegen. Die Linie selbst wird in den Zeilen 7 – 12 gesetzt. Dabei steht jeweils ein Koordinaten-Paar für einen Ankerpunkt, der die Linie startet, bzw. durch den sie verläuft. Im Vergleich zu Fabric.js ist dabei zur Zeichnung einer mehrteiligen Linie nur ein Objekt nötig. Soll die Linie erweitert werden, werden weitere Punkte in das Array geschrieben, wodurch ein Pfad beliebig viele Ankerpunkte haben kann.

Anders als Fabric.js verfügt Kinetic.js nicht über ein eigenes Objekt zur Generierung von Bézierkurven. Aus diesem Grund müssen die Kurven in der schon in Bézierkurven-Beispiel verwendeten Schreibweise notiert werden:

```
var stage = new Kinetic.Stage({
 container: 'kinetic_canvas',
 width: 500,
 height: 300
});
var layer = new Kinetic.Layer();
var shape = new Kinetic.Shape({
 drawFunc: function(context) {
       context.beginPath();
       context.moveTo(188, 130);
       context.bezierCurveTo(140, 10, 388, 10, 388,
170);
       context.lineWidth = 10;
```

61 Der DOM-Baum ist die Gesamtheit aller auf der Webseite vorhandenen Element e (Knoten).

```
            context.setAttr('lineWidth', 10);
            context.stroke();
        },
    });
    layer.add(shape);
    stage.add(layer);
```

Die ersten fünf Zeilen des Codes sind identisch zum Beispiel davor. Mit dem Objekt `Kinetic.Shape` wird ein freie Form initiiert, die mit der Funktion `drawFunc` in Zeile 8 den Code zur Definition der Bézierkurve enthält.

3.1.2 Zwischenfazit

Wie deutlich wird, vereinfacht die Verwendung einer „canvas“-Bibliothek die Arbeit mit dem „canvas“-Element, da der Entwickler direkt mit „canvas“-Objekten und deren Eigenschaften und Verhalten arbeiten kann. Besonders hilfreich ist dies bei der Arbeit mit vielen gleichzeitig auf dem „canvas“-Element zu rendernden Objekten, wie es bei komplexen HTML5-Apps wie Spielen und Dienstprogrammen gängig ist. In einem solchen Fall „lohnt“ sich der Aufwand, eine „canvas“-Bibliothek zu verwenden, da sie die leichte Verwaltung aller gezeichneten Objekte ermöglicht. Wird das „canvas“-Element jedoch nur zur Darstellung einfacherer statischer Grafiken verwendet, die jeweils einzeln und automatisiert aus einer Datenbank eingelesen werden, ist der Einsatz einer Bibliothek nicht unbedingt nötig. Dies liegt daran, dass der Aufwand zur Initialisierung einer „canvas“-Bibliothek bei Zeichnung verhältnismäßig einfacher statischer Grafiken proportional größer ist als die direkte Verwendung der in Kapitel 3.1 erläuterte „low level“ JavaScript-Code.

3.2 Konzeption eines Übersetzungs-Tools

Im Folgenden soll dargelegt werden, dass ein auf Webseiten eingebundenes Tool in der Lage sein kann, automatisiert einzelne schriftsprachliche Begriffe in Gebärden übersetzt darzustellen. Das Tool ist in erster Linie für Betreiber privater Webseiten gedacht, die damit in der Lage wären, die Barrierefreiheit für gehörlose Menschen ohne den Einsatz von Gebärden-Videos zumindest etwas zu verbessern. Nachfolgend wird skizziert, welchen Aufbau ein Übersetzungs-Werkzeug („*Tool*“) zur Darstellung von Gebärden haben könnte und wie es sich durch Nutzer bedienen lässt. Besonders die im Hintergrund ablaufenden Prozesse des Tools werden dabei dargestellt.

3.2.1 Benutzungs-Szenario

Die Verwendung des Tools durch einen gehörlosen Nutzer wird in folgendem Szenario beschrieben:

Ruft ein schwerhöriger oder gehörloser Mensch eine Webseite auf, auf der das Tool eingebunden ist, wird er von einer in einfachen Worten geschriebenen Einblendung am oberen Bildschirmrand darüber informiert, dass er Wörter im Text, deren Bedeutung er nicht versteht, mithilfe eines Übersetzers als Gebärde angezeigt bekommen kann. Er hat nun die Möglichkeit, das Tool zu aktiveren. Klickt der Nutzer ein Wort an, das er nicht versteht, öffnet sich eine Einblendung ähnlich einer Lightbox, welche das zu übersetzende Wort als Gebärde darstellt (siehe Abbildung 7). In dem Fenster wird zusätzlich das angefragte Wort angezeigt sowie die Möglichkeit zum Schließen des Einblendung gegeben. Wenn der Nutzer auf ein Wort klickt, das nicht in der Datenbank des Tools gespeichert ist, wird er mit einer dezenten Nachricht darüber informiert. Kehrt er zu einem späteren Zeitpunkt auf die Webseite zurück, hat sich das Tool gemerkt, ob er die Übersetzungs-Funktion benutzen möchte und aktiviert sie, bzw. bleibt inaktiv.

Ein grafischer Entwurf des Tools demonstriert die Funktionsweise anhand des Wikipedia-Eintrags zum Begriff „Tasse“. Rechts ist das geöffnete Fenster des Tools abgebildet, in dem der übersetzte Begriff in großer Schrift wiederholt wird. Darunter befindet sich die eigentliche Gebärde, die in diesem Entwurf noch aus einer Rastergrafik besteht. Der Button zum Schließen des Fensters befindet sich rechts oben im Fenster.

Abbildung 7: Entwurf des Tools (bearbeiteter Screenshot)

3.2.2 Schematischer Aufbau

In folgender Abbildung ist die prinzipielle Funktionsweise des Tools skizziert.

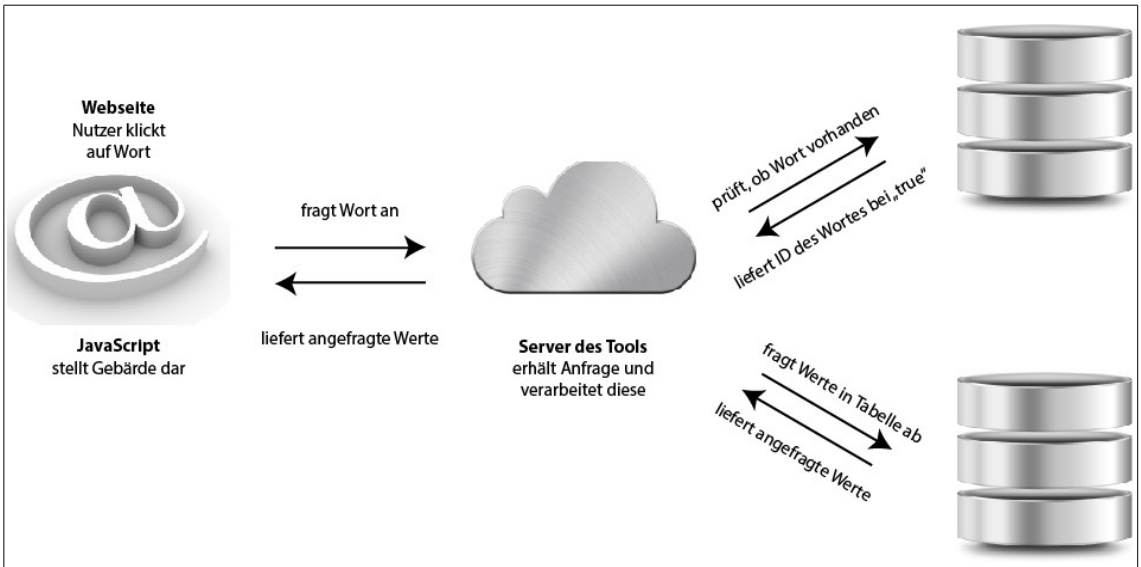

Abbildung 8: schematische Funktionsweise des Tools

Die Webseite, die das Tool eingebunden hat, sendet eine das angeforderte Wort enthaltende Anfrage an den Server des Tools. Dieser prüft in einer Datenbank, ob das Wort vorhanden ist. Ist dies der Fall, erhält er die ID des Begriffs und nimmt diese, um in einer zweiten Tabelle die zur Darstellung der Gebärde benötigten Werte abzufragen. Diese sendet er JSON-formatiert zurück an die Webseite, auf der das Tool die empfangenen Werte nimmt, um sie im „canvas“-Element darzustellen.

Warum sich das Tool die erforderlichen Daten von einem entfernten Server holt, hat zwei Gründe: Zum einen bleibt die Größe der Dateien klein, die auf dem Server der Webseite erforderlich sind. Zum anderen erhält der Nutzer so bei jedem Abruf den aktuellen Stand aller verfügbaren Gebärden und ist nicht darauf angewiesen, dass der Webseiten-Betreiber das Tool updatet.

Zur Interaktion mit den Nutzer sowie zur Zeichnung der Gebärden setzt das Tool auf der Webseite JavaScript ein. Dabei muss es folgende Schritte beherrschen:

- Überprüfung, ob Nutzer früher bereits die Webseite besucht hat
  - wenn ja: Prüfen, ob Nutzer Übersetzung aktiviert oder deaktiviert hat
  - wenn nein: Einblenden einer Leiste, die den Nutzer über die Verfügbarkeit der Übersetzung informiert
    - bei Klick auf Ja: Aktivierung der Übersetzungsfunktion, Setzen einer Erinnerung mithilfe der „Web Storage“-Technik („true“)
    - bei Klick auf Nein: Ausblenden der Leiste, Setzen einer Erinnerung („false“)
- Erkennen des ausgewählten Wortes
- Senden des Begriffs an den Server des Tools, Empfang der Antwort (AJAX)
- Öffnen der Lightbox, die das „canvas“-Element beinhaltet
- Parsen der empfangen Werte im „canvas“-Element, Zeichnung der Gebärde

Funktionsweise des Tool-Servers

Der Server des Tools ist in PHP objektorientiert programmiert und kommuniziert mit einer SQL-Datenbank mittels der „MySQLi“-Erweiterung. Die Datenbank selbst besteht aus zwei Tabellen, von denen „s_t_terms“[62] die Liste mit als Gebärde verfügbaren Worten enthält, während „s_t_sign_values“ die Werte der Gebärden für die einzelnen Begriffe speichert.

62 „s_t“ steht für **S**ign **T**ranslator.

s_t_term_id	s_t_term	s_t_notes
1	Aal	1
2	Ampel	-1
3	Auto	1
4	Autobahn	2
…	…	...

Tabelle 4: exemplarischer Aufbau der Tabelle "s_m_terms"

Die Tabelle besteht aus den drei Spalten „s_t_term_id“ (SMALLINT), „s_t_term“ (VARCHAR(32)) und „s_t_notes“ (TINYINT). Die erste Spalte enthält die fortlaufende Nummerierung der einzelnen Begriffe, die der Server abfragt, wenn der zu übersetzende Begriff vorhanden ist. „s_t_term“ enthält die verfügbaren Begriffe und „s_t_notes“ speichert optional Bemerkungen zu den Einträgen ab, die der Server einliest und als Meldung an das Tool weiterleitet, sofern gesetzt:

- nicht gesetzt: keine Meldung
- „1“ gesetzt: Wort besteht aus einer einzelnen Gebärde
- „2“ gesetzt: Wort besteht aus zwei zusammenhängenden Gebärden
- „-1“ gesetzt: Gebärde des angefragten Wortes fehlerhaft/unvollständig

Die simple SQL-Abfrage zur Abfrage der ID des Begriffes sieht folgendermaßen aus (vereinfacht):

```
SELECT s_t_term_id, s_t_notes FROM s_t_terms
WHERE s_t_term = $term;
```

Die Variable $term enthält den vom Tool angefragten Begriff.

Erhält der Server bei einer Anfrage eines Nutzers auf der Webseite eine ID aus der Tabelle „s_t_terms“ (die Gebärde ist also vorhanden), führt er eine weitere (vereinfachte) SQL-Query mit der erhaltenen ID in der zweiten Tabelle „s_t_sign_values“ aus, um die Werte der Gebärde abzufragen:

```
SELECT s_t_value FROM s_t_sign_values
WHERE s_t_term_id = $id;
```

Die Variable $id enthält den Wert, den die erste Tabelle „s_t_terms“ im vorherigen Schritt zurückgegeben hat.

s_t_term_id	s_t_term_value
1	140,10,388,10,388,170 27,140,444,265,9,51 141,12,390,12,390,172 13,153,523,481,184,73 205,49,181,62,202,196
2	403,358,297,44,413,384 96,216,314,495,58,39 394,111,425,433,15,326 466,117,392,277,237,382 441,443,415,445,205,138
3	28,291,430,383,238,173 282,160,167,360,276,333 410,411,330,182,98,50 170,331,224,256,418,38 105,449,112,256,50,258
4	42,134,373,344,459,393 265,106,22,285,29,154 384,399,448,441,359,44 179,64,409,428,461,288 365,276,83,174,452,120
…	…

Tabelle 5: exemplarischer Aufbau der Tabelle "s_m_sign_values"; die Werte sind nicht real

Die Spalte „s_t_term_id“ enthält die gleichen IDs wie die gleichnamige Spalte der Tabelle „s_t_terms“. Die Spalte „s_t_term_value“ (LONGTEXT) speichert die Werte der Gebärde.

Auf diese Art und Weise kann das Tool jedem gehörlosen Besucher die angefragte Gebärde schnell und zuverlässig anzeigen.

4 Fazit und Ausblick

Die vorliegende Arbeit hat den derzeitigen Stand der Entwicklung des Internets zu einem mehr und mehr barrierefreien Netz aufgezeigt. Dabei wurde deutlich, dass der Verzicht auf externe Plugins wie Flash und Java einer der wichtigsten Mittel zur Verbesserung der Barrierefreiheit für alle Mitglieder der Gesellschaft ist. Genauso ist auch der Einsatz von HTML5 zu bewerten; es wurde ersichtlich, dass Screenreadern die korrekte Wiedergabe von Inhalten einer Webseite durch die Verwendung von den die Inhalte beschreibenden Tags erleichtert wird. Besondere Beachtung sollten Webentwickler dabei den HTML-Elementen `nav`, `main`, `section`, `article`, `aside` und `footer` zukommen lassen. In Bezug auf die in der BITV geforderten Zurverfügungstellung von Untertiteln und Audio- und Video-Transkriptionen hat sich gezeigt, dass das HTML5-`track`-Element in Verbindung mit dem WebVTT-Format alle Anforderungen an Alternativtexte für multimediale Inhalte erfüllt, und der einzige Grund für Webentwickler, noch nicht mit diesem Element zu arbeiten, nur in der bislang nicht vollständigen Browser-Unterstützung zu finden ist.

Einzig schwerhörige und gehörlose Menschen mit mangelnden Lese- und Schreibkompetenzen bilden eine isolierte Bevölkerungsgruppe, für die es bislang nicht gelungen ist, technische Hilfsmittel zur Rezeption von Texten im Internet zu entwickeln, sodass die Alternative der Einsatz von Leichter Sprache und/oder Gebärden-Videos ist. Da gezeigt wurde, dass die Produktion von Gebärden-Videos mit hohen Kosten verbunden ist, ist die Verwendung solcher Videos abseits von Internetauftritten der Behörden unrealistisch. Um diesen Mangel an Barrierefreiheit zu „bekämpfen“, wurde ein automatisiert arbeitendes Übersetzungs-Programm technisch konzipiert, das geschriebene Worte in Gebärden überträgt. Dabei wurde deutlich, dass solch ein Tool verhältnismäßig simpel strukturiert sein kann und aus technischer Sicht nichts gegen den Einsatz solcher Übersetzer spricht. Was jedoch mit nicht zu unterschätzendem Aufwand verbunden ist, ist die Erstellung und Digitalisierung der einzelnen Gebärden an sich. Um diesen Prozess zu vereinfachen, müssen Möglichkeiten geschaffen werden, Gebärden effizient in großer Anzahl zu digitalisieren. Sehr interessante Ansätze werden dazu zurzeit in bei „Microsoft Research“ geschaffen. Dort entsteht ein Projekt, welches mit der „Kinect“-Kamera in der Lage ist, die menschliche Gebärden zu erkennen und

diese als Avatar darzustellen.[63] Eine solche automatisierte Übersetzung von Schriftsprache in visuell dargestellte Gebärden auf Webseiten würde die Einbindung von Gebärden-Videos überflüssig machen und einem großen Anteil der Gesellschaft endlich Zugang zu einem globalen Netzwerk an Informationen verschaffen.

63 Unter http://research.microsoft.com/en-us/collaboration/stories/kinect-sign-language-translator.aspx findet man weitere Informationen zu dem Projekt.

Literaturverzeichnis

AKAMAI, ohne Namen: Retail Web Site Perfomance - Consumer Reaction to a Poor Online Shopping Experience, online im Internet: http://www.akamai.com/dl/reports/Site_Abandonment_Final_Report.pdf, 2006, Stand: 09.06.2014

BACH, Heinz: Sonderpädagogik im Grundriß. Berlin 1975

BUNDESMINISTERIUM FÜR ARBEIT UND SOZIALES (Herausgeber): Begründung zur Barrierefreien-Informationstechnik-Verordnung, BITV 2.0. 2011, online im Internet: http://www.bmas.de/SharedDocs/Downloads/DE/PDF-Gesetze/begruendung-bitv-2.0.pdf?__blob=publicationFile, Stand: 12.06.2014

BUNDESMINISTERIUM FÜR ARBEIT UND SOZIALES (Herausgeber): Verordnung zur Schaffung barrierefreier Informationstechnik nach dem Behindertengleichstellungsgesetz (Barrierefreie-Informationstechnik-Verordnung - BITV 2.0). 2011, online im Internet: http://www.bmas.de/SharedDocs/Downloads/DE/PDF-Publikationen/a712a-bitv-2.0-barrierefrei.pdf?__blob=publicationFile, Stand: 12.06.2014

CUTLER, Blake: Firefox & Page Load Speed – Part II, 2010, online im Internet: http://blog.mozilla.org/metrics/2010/04/05/firefox-page-load-speed-%E2%80%93-part-ii/, Stand 09.06.2014

DEUTSCHER GEHÖRLOSEN-BUND E. V. (DGB) (Herausgeber): „Barrierefreie Informationstechnik-Verordnung“ (BITV) – Rechtsverordnung zu § 11 Behindertengleichstellungsgesetz – Eine Stellungnahme des Deutschen Gehörlosen-Bundes e. V. Online im Internet: http://www.kestner.de/n/verschiedenes/presse/2004/Stellungnahme%20zu%20BITV.pdf, Stand: 20.06.2014

DWORSKI, Anja: Was ist Leichte Sprache? Presseinformation des Vereins Netzwerk Leichte Sprache. Online im Internet: http://www.leichtesprache.org/downloads/Presse-information%20Netzwerk%20Leichte%20Sprache%2029.8.2013.pdf, Stand: 20.06.2014

KLOTZ, Stefan: Vergessene Zielgruppe des Internets? Gehörlose und Internet - Analyse der Bedürfnisse einer Zielgruppe und Schlussfolgerungen am Beispiel Berlins. Potsdam, 1998, online im Internet: http://www.taubenschlag.de/html/infos/ergebnis.htm Stand 05.06.2014

LINDEN, Greg: Make Data Useful, 2006, online im Internet: https://sites.google.com/site/glinden/Home/StanfordDataMining.2006-11-29.ppt, Stand 09.06.2014

LYNCH, Patrick et al.: Web Style Guide: Basic Design Principles for Creating Web Sites. New Haven, 2009

MARCOTTE, Ethan: Responsive Webdesign. New York, 2011

MILLER, Richard B.: Response time in man-computer conversational transactions. Proc. AFIPS Fall Joint Computer Conference Vol. 33, 1968

MOSER, Christian et al.: Barrierefreiheit im Internet - … einfach erklärt, online im Internet: http://www.einfach-barrierefrei.net/verstehen/hilfsmittel, Stand 05.06.2014

NIELSEN, Jakob: Designing Web Usability: The Practice of Simplicity. San Francisco 1998

NIELSEN, Jakob: Usability Engineering. New York, 1993

ZAYTSEV Juriy: Introduction to Fabric.js. Part 1. Online im Internet: http://msdn.microsoft.com/en-us/magazine/jj714178.aspx, Stand 14.05.2014.

Anlagen

JavaScript-Code des Tools

Hinweis: Der hier abgebildete Code greift der Einfachheit halber nicht auf den externen Servers des Tools, sondern auf eine im gleichen Verzeichnis abgelegte CSV-Datei mit den entsprechenden Gebärden-Werten zu.

sign_translator.js

```
$(function() {
          $('head').prepend('<link   rel="stylesheet"   type="text/css"
href="./canvas/incs/style.css">');
    console.log(localStorage.s_t_rememberSettings);
    if (!localStorage.s_t_rememberSettings) {
        // Infobar zeigen
        var infobar;
        $.get('./canvas/infobar.html', initialize)
    }
            if   (localStorage.s_t_rememberSettings   ===   true   ||
localStorage.s_t_rememberSettings === 'true') { // localStorage only
stores strings ATM...
                            $('body').prepend('<script   defer="defer"
src="canvas/sign_translator_core.js"></script>');
    }

    function initialize(infobar) {
        $('body').prepend(infobar);
        $('.s_t.infobar > button').click(function() {
            if ($(this).html() == 'Nein') {
                localStorage.s_t_rememberSettings = false;
                $('.s_t.infobar').toggleClass('s_t-hide');
                $('.s_t.infobar-small').toggleClass('s_t-hide');
            }
            else {
                localStorage.s_t_rememberSettings = true;
                $('.s_t.infobar').remove();
                $('.s_t.infobar-small').remove();
                            $('body').prepend('<script  defer="defer"
src="canvas/sign_translator_core.js"></<scr></scr>ipt>');
            }
```

```
        });
        $('.s_t.infobar-small > span').click(function() {
            $('.s_t.infobar').toggleClass('s_t-hide');
            $('.s_t.infobar-small').toggleClass('s_t-hide');
        });
    }
});
```

Die Datei „sign_translator.js“ wird in jede Webseite eingebunden, die die Übersetzungs-Funktion erhalten soll. Sie prüft zuerst, ob der Nutzer bereits die Aktivierung der Übersetzung zugestimmt oder sie abgelehnt hat. Bei angeforderter Aktivierung wird die Leiste eingeblendet, die den Nutzer über die Verfügbarkeit der Übersetzungs-Funktion informiert und gleichzeitig wird die Hauptdatei des Tools („sign_translator_core.js“) nachgeladen.

sign_translator_core.js

```
var requested_sign; // = 'tasse'; // angefordete Gebaerde (String)
var raw_values; // Werte aus entsprechender Datei (String)
var line_arr; // Array mit einzelnen Wertzeilen pro Index
var single_line; // einzelne Zeile mit Werten (String)
var polygon; // Array mit eigentichen Werten fuer Bezierkurven

var canvas;
var context;

$.get('./canvas/canvas.html', function(canvas) {
    $('body').prepend(canvas);
    // Canvas-Variablen
    canvas = document.getElementById('sign_canvas');
    context = canvas.getContext('2d');

    $("body").click(function() {
        // Gets clicked on word (or selected text if text is selected)
        var t = '';
                            if    (window.getSelection    &&    (sel    =
window.getSelection()).modify) {
            // Webkit, Gecko
            var s = window.getSelection();
            if (s.isCollapsed) {
                s.modify('move', 'forward', 'character');
                s.modify('move', 'backward', 'word');
                s.modify('extend', 'forward', 'word');
                t = s.toString();
                     s.modify('move', 'forward', 'character'); //clear
selection
            }
            else {
                t = s.toString();
            }
              } else if ((sel = document.selection) && sel.type !=
"Control") {
            // IE 4+
            var textRange = sel.createRange();
            if (!textRange.text) {
                textRange.expand("word");
```

```
            }
            // Remove trailing spaces
            while (/\s$/.test(textRange.text)) {
                textRange.moveEnd("character", -1);
            }
            t = textRange.text;
        }
        console.log(t);
        translate(t);
    });
});

function translate(requested_sign) {
    //umwandeln
         requested_sign  =  requested_sign.toLowerCase().replace('  ',
'').replace(',','').replace(/ä/g,"ae").replace(/ö/g,"oe").replace(/ü/g
,"ue").replace(/Ä/g,"Ae").replace(/Ö/g,"Oe").replace(/Ü/g,"Ue").replac
e(/ß/g,"ss");
    $.get('./canvas/hand_signs/'+requested_sign+'.csv', draw_canvas)
        .fail(function() {
            console.log('fail');
        });

    /* here's where the magic happens */
    function draw_canvas(raw_values) {
        // Zeilen aufsplitten und Anzahl erhalten
        line_arr = raw_values.split('\n');

        context.beginPath();

        // Schleife für jede gefundene Zeile durchlaufen
        for (var i = 0; i < line_arr.length; i++) {

            // einzelne Zeile in String umwandeln
            single_line = line_arr.slice(i,i+1).join();
            console.log('single_line: '+single_line);

            // einzelne Zeile wieder in Array aufsplitten
            polygon = single_line.split(', ');
            console.log('polygon: '+polygon);
```

```
            /* the canvas stuff */
            if (i == '0') context.moveTo(188, 130);
             context.bezierCurveTo(polygon[0],polygon[1],polygon[2],pol
ygon[3],polygon[4],polygon[5]);
            context.lineWidth = 10;
            console.log(i);
        }
        context.stroke();
    }
}
```

Die Datei „sign_translator_core.js“ wendet eine Funktion an, die bei Klick auf ein Wort dieses an eine zweite Funktion zur „Weiterbehandlung“ übergibt. Dabei wird die HTML-Struktur des Dokuments nicht verändert, was sicherstellt, dass der Einsatz des Tools sich nicht negativ auf andere Scripte oder das visuelle Erscheinungsbild der Webseite auswirkt.

Der gewählte Begriff wird von Umlauten, Leer- und Interpunktionszeichen befreit und dann genutzt, um in einem festgelegten Ordner nach einer CSV-Datei mit entsprechendem Dateinamen zu suchen. Existiert diese Datei, wird der Inhalt extrahiert, nach Zeilen aufgeteilt und für jede Zeile das Objekt `context.bezierCurveTo()` mit den entsprechenden Werten aufgerufen.

Weitere Dateien

Weitere Dateien enthalten das HTML, das zur Anzeige der Leiste, bzw. zur Erstellung des „canvas“-Elements benötigt werden und das CSS zur grundlegenden Gestaltung der Leiste.

Eigenständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe. Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

_____

Ort, Datum Vorname Nachname