

Ein JavaScript-Framework zur erweiterten Nutzung strukturierter und annotierter SVG-Grafiken in Webseiten

Christiane Taras, Michael Santoso, Thomas Schlegel, Thomas Ertl

Institut für Visualisierung und interaktive Systeme (VIS), Universität Stuttgart
{taras, thomas.schlegel, ertl}@vis.uni-stuttgart.de, michaelasantoso@gmx.net

Zusammenfassung

Die Verwendung von Scalable Vector Graphics (SVG) ist ein wichtiger Schritt zur Verbesserung der Zugänglichkeit von graphischen Informationen. SVG-Grafiken werden heute meist nur mit dem Ziel gestaltet, den richtigen visuellen Eindruck zu erzeugen. Auf Metainformationen, semantische Gruppierungen, Verknüpfungen und Annotationen wird kaum Wert gelegt. Dadurch entstehen trotz der enormen Potenziale von SVG häufig nur schwer zugängliche Grafiken. Dies begründet sich hauptsächlich darin, dass die Mehrarbeit, die die zugängliche Gestaltung von SVG-Grafiken bedeutet, in der statischen Darstellung auf Webseiten, wozu SVG-Grafiken heute hauptsächlich verwendet werden, keinen Mehrwert bringt. Dieses Paper stellt ein JavaScript-Framework vor, das es ermöglicht, interaktive Webseiten auf Grundlage von gut strukturierten und annotierten SVG-Grafiken zu gestalten. Dadurch sollen die Vorteile solcher Grafiken einer breiten Benutzergruppe zugänglich gemacht werden, um die Verbreitung solcher Grafiken und damit die Verbesserung der Zugänglichkeit von Grafiken in Webseiten insgesamt zu fördern.

1 Einleitung

Es ist unbestritten, dass Grafiken, die auf SVG (Scalable Vector Graphics) basieren, durch ihre Möglichkeiten zur Strukturierung und Annotation und die verlustlose Skalierbarkeit einen wesentlichen Beitrag zur Verbesserung der Zugänglichkeit von Grafiken und somit von Dokumenten, in denen sie eingesetzt werden, leisten können. Beispielsweise hat Kerstin Altmanninger mit Access2Graphics (Altmanninger & Wöß, 2008) gezeigt, wie aus SVG-Grafiken verschiedene Darstellungen für unterschiedliche Benutzergruppen erzeugt werden können. Martin Rotard entwickelte mit SVG4Blind (Rotard et al., 2004) ein Programm, das auch Blinden Zugang zu den grafischen Elementen von SVG-Grafiken ermöglicht.

Durch die zunehmend bessere Unterstützung von SVG in Webbrowsern finden immer mehr SVG-Grafiken Einzug in bedeutende Webseiten wie etwa Wikipedia. Hier werden sie wegen ihrer verlustlosen Skalierbarkeit und guten Änderbarkeit geschätzt. Betrachtet man diese Grafiken näher, so stellt man fest, dass sie häufig schlecht strukturiert und wenig annotiert sind. Der Grund hierfür ist offensichtlich. Weder eine gut durchdachte Struktur noch Annotationen haben einen Einfluss auf die visuelle Darstellung der SVG-Grafiken. Diese ist aber entscheidend für den Großteil der Webseiten-Ersteller und -Benutzer. Die Mehrarbeit, die gute Strukturierung und Annotationen bedeuten, bringt keinen Mehrwert im Einsatz.

Abbildung 1 zeigt zwei typische Beispiele für SVG-Grafiken aus dem Web (links und Mitte). Beide Grafiken sind für die visuelle Darstellung gut aufbereitet, besitzen aber keinerlei Annotationen. Auch die Strukturierung ist schlecht. In der Deutschlandkarte, die sogar für die Verwendung in Wikimedia empfohlen wird¹, sind nicht, wie man vermuten könnte, die einzelnen Bundesländer eigenständige Elemente der SVG-Grafik, sondern ergeben sich nur aus Grenzlinien, wie die hervorgehobenen Linien in der Abbildung zeigen. Diese sind nicht einmal so gestaltet, dass damit die Bundesländer jeweils einzeln dargestellt werden könnten. Sie erstrecken sich über mehrere Bundeslandgrenzen. In der Darstellung des Versuchsaufbaus (Mitte) wurde keinerlei Gruppierung auf die Elemente angewendet. Auch die Reihenfolge, in der die Elemente im XML-Code auftauchen, ist nicht sinnvoll, wie Abbildung 1 (rechts) zeigt.

Um die Verbreitung guter, das heißt sinnvoll strukturierter und annotierter, SVG-Grafiken zu fördern, müssen deren Vorteile für möglichst viele Benutzer deutlich werden. Ein Mittel dazu, ist die Gestaltung

¹ <http://commons.wikimedia.org/wiki/File:Deutschland.svg>

interaktiver Webseiten mit Hilfe von SVG-Grafiken. Dies macht das Potenzial von SVG einer stetig wachsenden Gruppe von Personen mit unterschiedlichsten Bedürfnissen zugänglich. So kann der Bedarf nach gut strukturierten und annotierten SVG-Grafiken gesteigert werden, was wiederum zu einer Verbesserung bei den SVG-Editoren führt. Um die Gestaltung solcher Webseiten zu erleichtern haben wir ein JavaScript-Framework entwickelt, das jedem Webseiten-Ersteller die Möglichkeit geben soll, gute SVG-Grafiken auf einfache Weise flexibel in Webseiten einzubinden und so deren Potenzial zu nutzen.

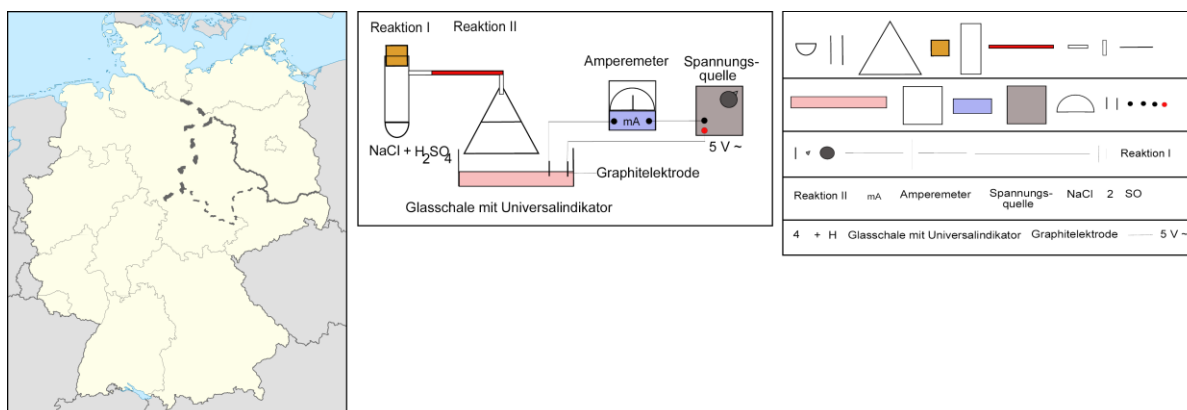


Abbildung 1: Beispiele für SVG. (links: Deutschlandkarte „Germany_location_map.svg“ von <http://commons.wikimedia.org/>, Mitte: Darstellung eines Versuchsaufbaus aus der Chemie von http://de.wikibooks.org/wiki/Anorganische_Chemie_für_Schüler:_Säure_-_Basen_-_Reaktionen, rechts: Darstellung der Elemente der mittleren Grafik in der Reihenfolge, in der sie in der Quelldatei auftreten).

2 Strukturierung und Annotation von SVG-Grafiken

Im Folgenden wird erläutert, welche Möglichkeiten SVG zur Strukturierung und Annotation bietet, welche Potenziale sich aus gut strukturierten und annotierten SVG-Grafiken ergeben und in wie weit Editoren und Webbrowser die Gestaltung und Darstellung solcher SVG-Grafiken derzeit von unterstützen.

2.1 Umsetzungsmöglichkeiten

Bei der Entwicklung von SVG wurde speziell darauf geachtet, dass ein Grafikformat entsteht, welches die Zugänglichkeit von Grafiken erhöht. Die Zielsetzung des World Wide Web Consortium (W3C) ist eindeutig: „Publish highly-structured documents, not just graphical representations“². Um dies auch möglich zu machen, wurden verschiedene Elemente in SVG definiert. Dazu gehören das Gruppierungselement (<g>) zur strukturellen Gruppierung von Elementen, Elemente zur Angabe von Beschreibung und Titel (<desc> und <title>) eines Elementes (nicht nur der Gesamtgrafik), sowie Elemente zur Wiederverwendung und Referenzierung anderer Elemente. Auch die Arbeit mit Standard-Formen wie Rechtecken und Kreisen, sowie die Verwendung von Cascading Style Sheets (CSS) zur Gestaltung der Grafik leisten erhöhen die Zugänglichkeit. Eine ausführliche Beschreibung, mit welchen Mitteln SVG-Grafiken zugänglich gestaltet werden können, wird vom W3C bereitgestellt (McCathieNevile & Koivunen, 2000).

Neben den dort aufgeführten Erläuterungen sollte auch beachtet werden, dass ein Element (wie beispielsweise ein Pfad) immer nur Elemente der Grafik enthält, die auch wirklich zusammen gehören. Nur so können diese Elemente sinnvoll beschrieben werden und nur so kann sich eine sinnvolle Struktur in der SVG-Grafik ergeben (vergleiche Abschnitt 1). Zudem ist der Nutzen von CSS zur Integration von semantischen Informationen in eine SVG-Grafik hervorzuheben. CSS-Klassen können verwendet wer-

² <http://www.w3.org/TR/2003/REC-SVG11-20030114/access.html>

den, um neben der strukturellen Gruppierung mit dem Group-Element weitere logische Gruppierungen vorzunehmen. So kann man beispielsweise in einer Deutschlandkarte einerseits alle Elemente (Städte, Kreise, etc.), die in einem Bundesland liegen, sowie das Bundesland selbst in ein Gruppierungs-Element eingliedern und trotzdem alle Städte über die CSS-Klasse „stadt“ gruppieren. Die CSS-Klasse ist also nicht nur ein Hilfsmittel für die Gestaltung der Elemente. Deshalb sollten hier sinnvolle Benennungen verwendet werden. Dies gilt auch für die IDs von Elementen, da diese für die Verknüpfung mit anderen Elementen genutzt werden können. Ein Pfeil-Symbol, das über das <use>-Element wiederverwendet wird, sollte also beispielsweise nicht „symbol1234“ heißen, sondern „pfeilsymbol“. Abbildung 2 (links) zeigt beispielhaft den Quellcode einer SVG-Grafik, die die genannten Elemente und Attribute benutzt.

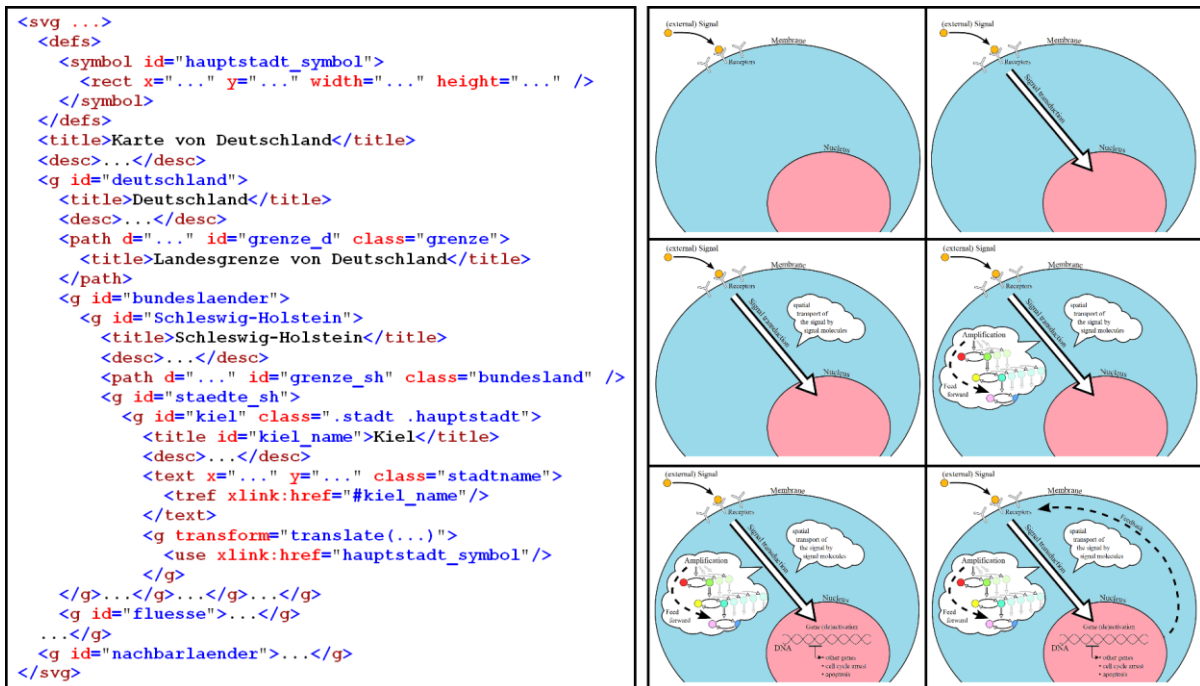


Abbildung 2: links: Quellcode einer gut strukturierten und annotierten SVG-Grafik, rechts: Schrittweiser Aufbau einer SVG-Grafik zur Signalausbreitung in Zellen (von Martin Falk, VIS)

2.2 Nutzungspotenzial

Die Zugänglichkeitsunterstützung von SVG wurde hauptsächlich mit dem Gedanken an Menschen entwickelt, die aufgrund einer Sehbehinderung die graphische Darstellung nicht erfassen können und nur mit textuellen Ausgaben arbeiten. Aber auch für andere Nutzer kann sie sehr hilfreich sein.

Annotationen von Grafiken tragen allgemein zum Verständnis einer Grafik bei. Eine komplexe Grafik wie beispielsweise die Darstellung der Signalausbreitung in Abbildung 2 (rechts unten) ist ohne zusätzliche Beschreibung kaum verständlich. Wird die Beschreibung direkt in das Bild integriert, wie es über die Elemente <title> und <desc> in SVG möglich ist, so steht sie jedem zur Verfügung, an den die Grafik-Datei weitergegeben wird. Es ist keine zusätzliche Datei nötig, somit kann die Beschreibung auch nicht verloren gehen. Zusätzlich kann die Annotation beispielsweise bei Integration der Grafik in einer Webseite als Teil des Webseitentextes, als Tooltip oder auch auditiv ausgegeben werden. Dabei könnte die Ausgabe mit einem Highlight in der Grafik oder dem Ein- und Ausblenden von Elementen verknüpft werden. Durch die Möglichkeit, SVG-Grafiken mit JavaScript zu beeinflussen, können diese Aktionen auch mit Benutzerinteraktionen verknüpft werden, um die SVG-Grafik dynamisch zu verändern. Somit können leicht verschiedene Aufnahmekanäle angesprochen werden, was zu einem besseren Verständnis des Inhaltes führt, den die Grafik vermitteln will. Besonders für E-Learning ist dies interessant.

Das Ein- und Ausblenden oder Hervorheben von Elementen passend zu ihren Beschreibungen ist natürlich nur möglich, wenn die Elemente den Annotationen auch zuordenbar sind und einzeln in ihrer Darstellung verändert werden können. Wenn die Grafik also gut strukturiert ist. Sind die Elemente zusätzlich sinnvoll geordnet, so kann sie, wie in Abbildung 2 gezeigt, Stück für Stück dargestellt werden. Auch dies kann das Verständnis der Grafik fördern. Durch den schrittweisen Aufbau kann die Aufmerksamkeit des Betrachters nach und nach auf die einzelnen Bestandteile gelenkt werden, die für sich eventuell leichter zu erfassen sind. Und auch für eine textuelle Ausgabe ist eine sinnvolle Reihenfolge natürlich wichtig.

Für die taktile Ausgabe ist ein schrittweiser Aufbau auch bei einfachen Grafiken essentiell, da es bei der taktilen Erfassung wesentlich schwieriger ist, einen Gesamteindruck zu gewinnen. Hier spielt auch die Verwendung von Standardformen und aussagekräftigen class- und id-Attributen eine große Rolle. Mit einem Hinweis, was die zu erfassende Form darstellt, kann sich der Nutzer leichter ein mentales Bild davon machen. So ist es wesentlich einfacher diese Form auch zu ertasten. Auch bei rein textueller Ausgabe ist es für den Benutzer so einfacher sich die graphische Darstellung vorzustellen.

Die taktile Erkennung und mentale Vorstellung einer Grafik wird ebenfalls durch die Wiederverwendung von Elementen der Grafik durch Referenzierung erleichtert. Das mehrfach verwendete Element muss nur einmal erfasst werden. Bei folgenden Verwendungen ist es bereits bekannt. Zudem erleichtert dies auch die Erstellung und Änderung der Grafik und bringt somit auch direkte Vorteile für den Grafik-Designer. Das mehrfach verwendete Element muss nur einmal erstellt werden und bei Änderungen muss nur dieses eine Element bearbeitet werden. Außerdem reduziert die Wiederverwendung den XML-Code, wodurch die Bearbeitung der Grafik auch für Benutzer von nicht-graphischen Editoren erleichtert wird. Ähnliches gilt auch für die Verwendung von Stylesheets anstatt Inline-Styling für die Gestaltung der Grafik.

2.3 Unterstützung durch Editoren und Browser

Tabelle 3 zeigt eine Übersicht, welche Zugänglichkeits-Elemente von SVG von aktuellen Editoren unterstützt werden. Es ist deutlich zu sehen, dass hier noch großer Verbesserungsbedarf besteht.

	Adobe Illust- rator CS2 ³	Amaya v11 ³	GLIPS Gra- fitti v1.5 ³	Inkscape v0.46 ³	OpenOffice 3.0 Draw ³	Sketsa SVG Editor v6.0 ³
verschachtelte Gruppierung	Ja	Ja	Ja	Ja	Ja	Ja (verschachtelt nur XML ⁴)
Basic Shapes (<rect>, <line> etc.)	Ja	Ja	Ja	nur <rect>, Rest als Path	Nein (nur als Path)	Ja
Titel und Beschreibung erstellen	Nein	XML ⁴	Nein	XML ⁴	Nein ⁵	XML ⁴
CSS-Klassen	Ja	XML ⁴	Nein	XML ⁴	Nein	XML ⁴
IDs zuweisen	Nein	XML ⁴	Ja	Ja	Ja	Ja
Text als <text>	Ja	Ja	Ja	Ja	Ja	Ja
<use> nutzen	Nein	XML ⁴	Nein	Ja	Nein	XML ⁴
<tref> nutzen	Nein	XML ⁴	Nein	XML ^{4,6}	Nein	XML ⁴
<glyph> nutzen	Nein	XML ⁴	Nein	XML ⁴	Nein	XML ⁴

Tabelle 3: Unterstützung ausgewählter SVG-Elemente in aktuellen SVG-Editoren

³ <http://www.adobe.com/svg/illustrator/illustrator.html>, <http://www.w3.org/Amaya/>, <http://glipssvgeditor.sourceforge.net/>, <http://inkscape.org/>, <http://www.openoffice.org/product/draw.html>, <http://www.kiyut.com/products/sketsa/>

⁴ Über integrierten XML-Editor

⁵ Eingabe wird nicht gespeichert

⁶ Referenzierter Text wird aber nicht im Bild angezeigt

Zwar besitzen einige Editoren einen integrierten XML-Editor, über den quasi jedes Element eingefügt werden kann. Dies kann aber nicht als wirkliche Unterstützung gewertet werden, da die meisten Grafik-Designer nicht zum XML-Editor greifen werden. Zudem werden spezielle Elemente wie `<use>` und `<ref>` zur Referenzierung von Grafik-Elementen beziehungsweise Textelementen nur dann genutzt werden, wenn sie vom Editor quasi im Hintergrund angewendet werden, also sich der Grafik-Designer nicht direkt mit dem XML-Code auseinandersetzen muss. Ohne eine gute Unterstützung durch Editierprogramme wird sich die Qualität der im Web eingesetzten SVG-Grafiken auch langfristig nicht verbessern.

Von den Web-Browsern wie Firefox⁷, Opera⁷ und Internet Explorer (über Adobe SVG Viewer Plug-In⁷) werden die SVG-Zugänglichkeits-Elemente bereits fast vollständig unterstützt. Lediglich die Referenzierungs-Elemente (`<use>` und `<ref>`) sind noch nicht gänzlich umgesetzt.

3 Das Framework

3.1 Konzept und Technologie

Wir haben ein JavaScript-Framework entwickelt, mit dem sich leicht SVG-Grafiken in Webseiten einbinden und flexibel verwenden lassen. Das Framework ermöglicht es, die Vorteile gut strukturierter und annotierter SVG-Grafiken zu nutzen und so das Potenzial von SVG auszuschöpfen.

Ein bedeutender Punkt unserer Entwicklung ist, dass die Grafik entsprechend dem Model-View-Controller-Prinzip getrennt von der Interaktionsschicht existiert. In die SVG-Grafik selbst müssen keine JavaScript-Funktionen eingebunden werden. Somit kann ein und dieselbe Grafik ohne Änderungen in verschiedenen Webseiten mit verschiedenen Interaktionen genutzt werden. Beispielsweise kann eine SVG-Deutschlandkarte an einer Stelle zur Darstellung der Aufteilung Deutschlands in Bundesländer dienen und an einer anderen Stelle zur Darstellung der Flüsse oder Landschaften. Der Grafikdesigner muss nur eine Grafik erstellen. Der Webseiten-Designer kann diese dann flexibel in verschiedenen Webseiten einsetzen beziehungsweise Teile von Webseiten aus den Texten der SVG-Grafik generieren lassen. Die Interaktionen werden über Scripte in der Webseite durch das Framework in die SVG-Datei eingeschleust.

Das Framework benutzt die JavaScript-Bibliothek „jQuery“ (Resig et al., 2009) und deren Plugin „jQuery SVG“ (Wood, 2008) auf und arbeitet über die vom W3C definierten DOM-Schnittstellen (W3C, 2003). Dadurch kann es problemlos in verschiedenen Browsern und mit verschiedenen SVG-Viewern verwendet werden. Außerdem kann es mit weiteren JavaScript-Bibliotheken verknüpft werden, um beispielsweise ansprechende und komplexe Tooltips, wie sie von der JavaScript-Tooltips-Bibliothek von Walter Zorn (Zorn, 2008) erzeugt werden, zu integrieren (siehe Abbildung 4). Auf die Integration bestehender Bibliotheken wird bei der Entwicklung des Frameworks großer Wert gelegt, um einen Beitrag zur Verknüpfung der vielfältigen Entwicklungen, die im Internet kursieren, zu leisten.

Ein wichtiges Ziel unserer Arbeit war es, dass das Framework auch mit wenig Wissen über JavaScript leicht einzusetzen ist. Der Benutzer muss sich weder um die Erstellung der für die Änderungen am SVG oder die Interaktion mit dem SVG nötigen Elemente, noch um die Ereignisbehandlung oder die Verknüpfung zwischen Elementen und Ereignisbehandlung kümmern. All dies ist im Framework gekapselt und geschieht automatisch. Trotzdem lässt das Framework einem Benutzer mit guten Kenntnissen in JavaScript durch optionale Parameter in den Funktionsaufrufen viel gestalterische Freiheit. Zudem war es uns wichtig, dass die Webseite in gewohnter Weise durch CSS gestaltet werden kann. Deshalb arbeiten alle Funktionen auf einem Skelett der Webseite. Werden Elemente in der Webseite erzeugt, so werden diese immer in vorgegebene Elemente eingefügt.

Sofern die Funktionen des Frameworks individueller verwendet werden als mit dem Standard-Verhalten (also mit Angabe von optionalen Parametern), sind Kenntnisse über den Aufbau der zu verwendenden

⁷ https://developer.mozilla.org/en/SVG_in_Firefox, <http://www.opera.com/docs/specs/opera95/#graphics>,
<http://www.adobe.com/svg/indepth/pdfs/CurrentSupport.pdf>

SVG-Datei natürlich unumgänglich. Auch hier zählt sich eine gute Strukturierung und Annotation aus. Je besser die SVG-Datei gestaltet ist, desto einfacher ist sie zu erfassen und zu verwenden.

Soll das Framework für interaktive Webseiten eingesetzt werden, muss beim Benutzer JavaScript aktiviert sein. Ohne JavaScript ist keine Interaktivität möglich. Sollen lediglich Webseiten verwendet werden, deren Inhalt über das Framework generiert wurde, die aber für den Benutzer keine Beeinflussungsmöglichkeiten bieten, so kann das Framework auch ohne aktiviertes JavaScript beim Benutzer verwendet werden. Die Webseiten können über das Framework generiert und als neue Webseite gespeichert werden, die dann im Web zur Verfügung gestellt wird. Im Allgemeinen bringt die grundsätzliche Deaktivierung von JavaScript heutzutage aber viele Nachteile im Web und sollte deshalb überdacht werden.

3.2 Funktionalitäten und Verwendung

Das Framework bietet vielfältige Funktionen zur Manipulation und interaktiven Einbindung von SVG-Grafiken in Webseiten. Dazu gehören:

- Einfügen von Titel und Beschreibung beliebiger Elementen an beliebigen Stellen
- (verdeckungsfreies) Hervorheben von Elementen der SVG-Grafik oder der Webseite
- Änderung von Darstellungsattributen von Elementen der SVG-Grafik oder der Webseite
- Ein- und Ausblenden und Zoomen von Elementen der SVG-Grafik
- Darstellung von Teilen einer SVG-Grafik in einem anderen Bereich
- Kombination mehrerer SVG-Grafiken und Einfügen neuer Elemente in SVG-Grafiken
- Generierung von Eingabedialogen mit Auswertungsfunktion
- Erzeugung von Tooltips
- Generierung von Formularelementen zur Verknüpfung mit anderen Funktionen
- Verknüpfung der genannten Funktionen mit Maus- und Tastatur-Interaktion

Die Verwendung einiger dieser Funktionen soll an zwei Beispielwebseiten erläutert werden. Die Abbildung 4 und 5 zeigen Ausschnitte aus den Webseiten und dem zugehörigen Quellcode.

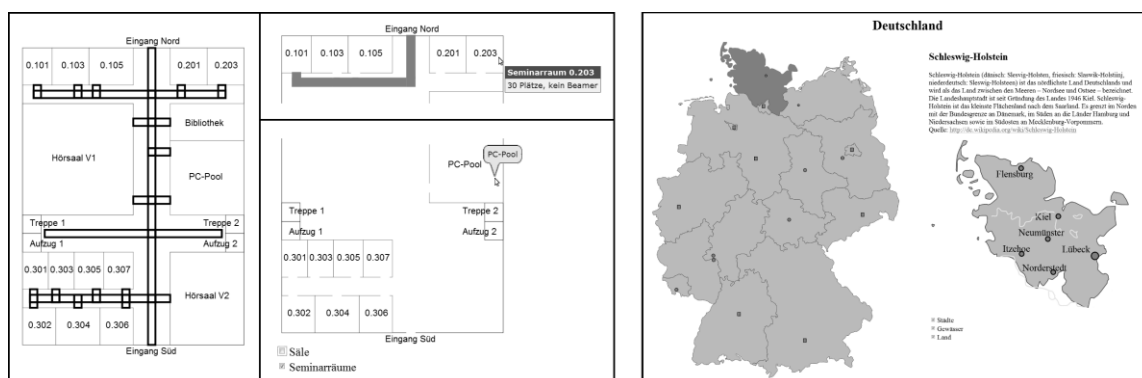


Abbildung 4: interaktive Anwendungsbeispiele des Frameworks
(links: Ausschnitte aus einer Raumplandarstellung, rechts: eine Deutschlandkarte).

Die Einbindung von SVG-Grafiken erfolgt, indem in der Webseite <div>-Elemente angelegt werden, die die CSS-Klasse „svg“ besitzen und den Pfad der zu ladenden SVG-Datei als Text enthalten. Beim Laden der Webseite sucht das Framework automatisch nach diesen <div>-Elementen und integriert die entsprechenden SVG-Dateien als Inline-SVG in die Webseite. Verwenden die SVG-Grafiken zur Angabe des

sichtbaren Bereichs das Attribut „viewbox“, so werden diese automatisch an die Größe des <div>-Elements eingepasst. Nachdem alle SVG-Grafiken geladen wurden, erfolgt automatisch der Aufruf der Funktion „afterSVGsLoaded“. In dieser Funktion müssen alle Aufrufe getätigt werden, die zur Manipulation der SVG-Grafiken und zum Einschleusen beziehungsweise Verknüpfen von Interaktionen dienen.

So kann beispielsweise in einem Raumplan (Abbildung 4 und 5 links) über den Aufruf „addTooltipDiv“ veranlasst werden, dass bei Überfahren von Elementen in der SVG-Grafik, die <title> und/oder <desc> als Kindknoten besitzen, mit der Maus automatisch ein Tooltip angezeigt wird. Der Tooltip ist entsprechend dem Element mit der ID „tooltip“ gestaltet und enthält die Inhalte aus den <title>- und <desc>-Knoten des SVG-Elements in den Abschnitten mit den IDs „tooltip_title“ bzw. „tooltip_desc“ (siehe Abbildung 4, linkes Bild, Abschnitt rechts oben). Über den Aufruf „generateCheckboxes“ werden automatisch Checkboxes in das <form>-Element mit der ID „checkform“ eingefügt, über die die direkten Kindgruppen des SVG-Wurzel-Elementes ein- und ausgeblendet werden können. Zur Beschriftung der Checkboxes werden die <title>-Knoten ausgelesen. Will man andere Elemente als die direkten Kindgruppen mit den Checkboxes beeinflussen, so kann man CSS-Selektoren für diese Elemente als optionalen Parameter angeben (siehe Abbildung 5 rechts). Außerdem können auch die Beschriftungen und sogar die Aktionen, die bei An- und Abwahl einer Checkbox ausgeführt werden, über optionale Parameter konfiguriert werden. Für eine noch individuellere Gestaltung der Webseite kann auch auf die Generierung der Checkboxes verzichtet werden. Checkboxes, die bereits im Webseitencode enthalten sind, können auch konventionell durch die Event-Attribute mit Funktionen aus dem Framework verknüpft werden.

<pre> <html><head> ... <script type="text/javascript" src="vis_svg_framework.js"/> <script type="text/javascript"> function afterSVGsLoaded() { addTooltipDiv("#raumplan"); generateCheckboxes("#raumplan"); highlight("#raumplan", ["#lo1, #lo2, #lo4, #lo6, #h1"], "red"); } </script> </head> <body> <h1>Raumplan Erdgeschoss</h1> <div class="svg" id="raumplan" > Raumplan.svg </div> <div id="tooltip"> <div id="tooltip_title"/> <div id="tooltip_desc"/> </div> <form name="checkform" action=""/> </body></html> </pre>	<pre> <html><head> ... <script type="text/javascript" src="vis_svg_framework.js"/> <script type="text/javascript"> function afterSVGsLoaded() { showTitleAndDesc("#deutschland", "#landname", "#landdesc", ["#laender g"], "click"); showPartIn("#deutschland", "#bundesland", ["#laender g"], "click"); generateCheckboxes("#bundesland", checkform, [".stadt", ".wasser", ".land"], ["Städte", "Gewässer", "Land"]); highlight("#deutschland", ["#laender g"], "blue", "mouseover"); } </script> </head> <body> <h1>Deutschland</h1> <div class="svg" id="deutschland"> Deutschland.svg </div> <div> <h2 id="landname"/> <p id="landdesc"/> <div id="bundesland"/> <form name="checkform" action=""/> </div> </body></html> </pre>
--	--

Abbildung 5: Quellcode für die Verwendungsbeispiele (links: Raumplan, rechts: Deutschlandkarte).

Der letzte Aufruf im Raumplan-Beispiel („highlight“) bewirkt, dass die Elemente, die durch die Liste im zweiten Parameter angegeben sind, mit der Farbe Rot hervorgehoben werden. Dadurch wird der Weg zum Raum „0.101“ markiert. Dies ist möglich, da im Raumplan Wegelemente eingebaut wurden, die

nicht über Wegkreuzungen hinweg verlaufen (Abbildung 4 links). Durch diese wohlüberlegte Anordnung der kann jeder Wegabschnitt einzeln hervorgehoben werden. Das Hervorheben kann natürlich durch Interaktionen wie die Auswahl eines Raumes aus einer Raumliste ausgelöst werden.

Das Beispiel zur interaktiven Deutschlandkarte (Abbildung 5 rechts) zeigt zusätzlich, wie Titel und Beschreibung von Elementen dynamisch in die Webseite eingefügt werden können. Durch den Aufruf „showTitleAndDesc“ werden bei Mausklick („click“) auf ein Gruppen-Element unterhalb des Elements mit der ID „laender“ (also auf ein Bundesland), Titel und Beschreibung der Gruppe in die Elemente mit den IDs „landname“ bzw. „landdesc“ eingefügt. Die Gestaltung als <h2>- bzw. <p>-Element bleibt dabei natürlich erhalten. Zusätzlich wird bei der gleichen Aktion das ausgewählte Bundesland separat und vergrößert im Element mit der ID „bundesland“ dargestellt. Mit Hilfe einer Template-Funktion des Frameworks wäre es auch möglich bei Aufruf der Webseite sofort die entsprechenden Informationen für alle Bundesländer in der vorgegeben Gestaltung anzuzeigen. Über die Integration von Eingabedialogen, die ebenfalls vom Framework unterstützt wird, könnte hier auch leicht eine interaktive Lerneinheit erzeugt werden, bei denen die Lernenden beispielsweise Namen von Städten und Flüssen in Deutschland oder die Zuordnung der Hauptstädte lernen können. Die Eingaben können automatisch mit den passenden Beschriftungen in der SVG-Grafik verglichen werden, welche vorher ausgeblendet werden können.

4 Fazit und Ausblick

Wir haben gezeigt, wie vielfältig SVG-Grafiken in Webseiten genutzt werden können und welche Vorteile durch Annotationen und eine gute Struktur auch für „normale Benutzer“ entstehen können. Es ist offensichtlich, dass die in Abschnitt 3 vorgestellten Funktionen nur mit gut strukturierten und annotierten SVG-Grafiken sinnvoll umgesetzt werden können. Die Erstellung solcher Grafiken erfordert zwar zunächst einen höheren Aufwand. Dieser zahlt sich aber aus. Unterstützungssysteme für Blinde und Sehbehinderte können die Grafik leichter in einer angepassten Weise präsentieren. Und mit Hilfe unseres Frameworks können mit wenig Aufwand interaktive Webseiten aus der Grafik erzeugt werden. So kommen die Vorteile solcher Grafiken auch anderen Benutzergruppen zu Gute. Damit wollen wir dazu beitragen, dass eine größere Nachfrage nach gut annotierten und strukturierten SVG-Grafiken entsteht und dadurch auch die diesbezügliche Unterstützung in SVG-Editoren ausgebaut wird. Nur so ist es möglich, dass mehr Webseiten mit guten SVG-Grafiken angereichert werden und so die Zugänglichkeit von graphischen Informationen in Webseiten und anderen Dokumenten erhöht wird.

Das Framework wird über die Webseite <http://www.vis.uni-stuttgart.de/~taras/SVGFramework> bereitgestellt und ständig weiterentwickelt. In Zukunft werden wir auch Funktionalitäten zur interaktiven Darstellung von MathML und zur Verknüpfung von MathML mit SVG integrieren, um auch hier die Potenziale aufzuzeigen und so die Verbreitung von zugänglichen Formeldarstellungen zu fördern.

5 Literaturverzeichnis

- Altmanninger, K. & Wöß, W. (2008): Accessible Graphics in Web Applications: Dynamic Generation, Analysis and Verification. In ICCHP'08: Proceedings of the 11th international conference on Computers Helping People with Special Needs, 378-385
- Resig, J. et al. (2009): jQuery: The Write Less, Do More, JavaScript Library. <http://jquery.com/> (23.01.2009)
- Rotard, M. et al. (2004): Exploring Scalable Vector Graphics for Visually Impaired Users. In ICCHP'04: Proceedings of the 9th international conference on Computers Helping People with Special Needs, 725-730
- Wood, K. (2008): jQuery SVG. <http://keith-wood.name/svg.html> (23.01.2009)
- Zorn, W. (2008): Tooltips per JavaScript / DHTML. <http://www.walterzorn.de/tooltip/tooltip.htm> (23.01.2009)
- McCathieNevile, C. & Koivunen, M. (2000): Accessibility Features of SVG – W3C Note 7 August 2000. <http://www.w3.org/TR/SVG-access/> (26.01.2009)
- World Wide Web Consortium (W3C) (2003): SVG Document Object Model (DOM) (14. Januar 2003). <http://www.w3.org/TR/SVG/svgdom.html> (26.01.2009)