Integration von Flash und Ajax

Ein interaktiver Messeplan

Projektarbeit von:
Tobias Dinter 167146
Tobias Erdrich 167150
Yael Bar-Zeev 167129
David Maus 167185

Betreuer:
Prof. Dr. Roland Riempp
Prof. Dr. Tom Rüdebusch

FB Medien- und Informationswesen
Hochschule Offenburg

Inhaltsverzeichnis

1. Einleitung

1.1 Aufgabe

Die vorliegende Arbeit entstand im Rahmen der Bachelor Projektarbeiten an der Hochschule Offenburg im Fachbereich Medien und Informationswesen im Wintersemester 2008/2009. Die Beschreibung der Projektarbeit lautete wie folgt:
„Die Kombination von Flash und AJAX bietet interessante Möglichkeiten bei der Gestaltung von dynamischen, interaktiven, multimedialen Webpräsenzen. Beide Technologien können sich dabei sinnvoll ergänzen und jeweils einen spezifischen Beitrag zur Gesamtwirkung beisteuern. Sinn der Projektarbeit ist es, das Potential der Kombination von Flash und AJAX zunächst zu eruieren und auf der Basis der dabei gewonnenen Erkenntnisse im zweiten Schritt eine prototypische Implementierung zu erstellen, die die Möglichkeiten in anschaulicher Weise präsentiert, so dass auch für Nicht-Fachleute die Vorteile und Potentiale offensichtlich werden."
Diese Arbeit ist in einen Theorieteil und in einen Praxisteil gegliedert. Im Theorieteil sollen die Hintergründe und Potenziale von AJAX und Flash näher beleuchtet und miteinander verglichen werden. Der Praxisteil beschreibt die Projektdefinition und Konzeption der Projektarbeit sowie die wichtigsten Funktionen und Objekte in ihrer technischen Umsetzung.

1.2 Rich Internet Applications

Kein Medium verbreitet sich schneller als das Internet. In den letzten sieben Jahren hat sich der Anteil der Nutzer von 37 Prozent der Bevölkerung ab 14 Jahre auf 60,2 Prozent erhöht. Parallel dazu ist der Anteil der Nichtnutzer von über der Hälfte der Bevölkerung auf etwa ein Drittel gesunken. 38,6 Millionen bundesdeutsche Erwachsene sind inzwischen online[1].

Im Zeitalter des „Web 2.0" geht der Trend immer mehr dahin, seinen Arbeitplatz oder persönlichen Anwendungen in einer Web-Applikation einzurichten, sodass auf diese überall via Internet zugegriffen werden kann. Daraus folgt, dass traditionelle Desktopanwendungen durch Web-Anwendungen ersetzt werden müssen. Um dies zu ermöglichen bedarf es einer neuen Art von Web-Applikation mit reichhaltigen Benutzeroberflächen, so genannte „Rich Internet Applications" (RIA). Sie verbessern das Benutzererlebnis durch umfangreiche Interaktionsmöglichkeiten und einem flüssigen und unterbrechungsfreien Arbeitsablauf.

Dies wird vor allem dadurch realisiert, indem viele Funktionalitäten, die in herkömmlichen Web-Anwendungen vom Server übernommen werden, auf den Client verlagert und direkt im Browser interpretiert werden. RIAs benötigen daher moderne Browser und leistungsfähige Client-Rechner, um mit der höheren Ressourcenbelastung fertig zu werden. Angesichts der steigenden Anzahl an Breitband-Verbindungen und leistungsfähigeren Computerprozessoren sollte eine echtzeitähnliche Interaktion mit der Anwendung jedoch problemlos funktionieren.

1 Bundesministerium für Wirtschaft und Technologie

RIAs können in unterschiedlichen Bereichen zum Einsatz kommen. Ein gutes Beispiel für durchdachte RIAs sind Geografische Kartensysteme wie „Google Maps" und „Yahoo Maps". Online Produktkonfiguratoren wie die auf den Seiten von „mini.de" oder „harley-davidson.com" sind ebenfalls zwei überzeugende Beispiele für RIAs. Gleiches gilt auch für Autorensoftware wie „Google Docs & Spreadsheets". Eine immer weiter anwachsende Zahl an Unternehmen hat mittlerweile RIAs im Einsatz, und auch in Zukunft werden immer mehr Anwendungsentwicklungsprojekte RIA-Techniken nutzen.

2. Theorieteil

2.1 AJAX - Eine Einführung

„AJAX" ist eigentlich keine neue Technologie, sondern setzt sich vielmehr aus bereits bestehenden Technologien zusammen, die ineinander greifen. „AJAX" bildet somit einen Oberbegriff für die Verwendung von:

- JavaScript und dem Document Object Model (DOM),
- XML (eXtensible Markup Language),
- dem JavaScript Objekt ***„XMLHttpRequest"*** (XHR)

„AJAX" ermöglicht flüssige Arbeitsabläufe und interaktive Funktionalitäten wie Drag & Drop oder aufklappende Kontextmenüs und sorgt dafür, dass diese Anwendungen im Internet nicht nur wie Desktop-Programme aussehen, sondern sich auch so anfühlen. Im Mittelpunkt steht dabei hauptsächlich die asynchrone Datenübertragung zwischen einem Server und einem Client (Browser).
Diese Form der Datenübertragung existiert bereits seit 1998, als Microsoft versuchte das E-Mail- und Organisationsprogramm „Outlook" im Internet Explorer nachzubilden. Das ständige Neuladen der Oberfläche umging Microsoft indem sie die Technologie „XMLHTTP" erfanden, und als ActiveX-Steuerelement[2] im Internet Explorer implementierten. Damit war es möglich den Server zu kontaktieren ohne die Seite komplett neu zu laden. Mittlerweile haben die meisten modernen Browser diese Technologie als eigenständiges (natives) XMLHttp-Request-Objekt implementiert.

2.1.1 Synchrone und asynchrone Datenübertragung

In diesem Kapitel soll geklärt werden, inwieweit sich das klassische synchrone Übertragungsmodell vom asynchronen Modell der Datenübertragung unterscheidet und welche Vorteile für den Benutzer und die Web-Anwendungen daraus resultieren.

Bei der synchronen Datenübertragung wird bei jedem Klick auf einen Link, oder beim Absenden eines Formulars vom Client eine Verbindung zum Server aufgebaut und ein HTTP-Request geschickt. Der Server antwortet mit der angeforderten Webseite, die er mit dem HTTP-Response an den Client zurücksendet. Während dieser Zeit ist der Arbeitsfluss des Nutzers unterbrochen, und er muss warten

2 ActiveX ist eine vorkompilierte Softwarekomponente, die als aktiver Inhalt eingesetzt werden kann

bis die HTML-Seite komplett auf dem Client eingetroffen ist.
Das AJAX-Konzept ermöglicht es dem Nutzer trotz einer gestellten HTTP-Anfrage weiter mit der Seite arbeiten zu können, ohne auf die Antwort vom Server warten zu müssen. Er kann sogar weitere HTTP-Anfragen stellen, ohne seinen Arbeitsfluss zu unterbrechen. Die Kommunikation mit dem Server läuft dabei im Hintergrund und die Seite wird nicht jedes mal komplett vom Server zum Client geschickt, sondern die benötigten Daten werden durch den Client sukzessive vom Server nachgefordert. Wenn die Antwort des Servers zur Verfügung steht, wird sie verarbeitet und auf der aktuell angezeigten Seite integriert. In der Abbildung unten wird der Arbeitsfluss einer herkömmlichen Web-Anwendung mit dem einer asynchronen Anwendung, welche die AJAX-Engine integriert hat, verglichen.

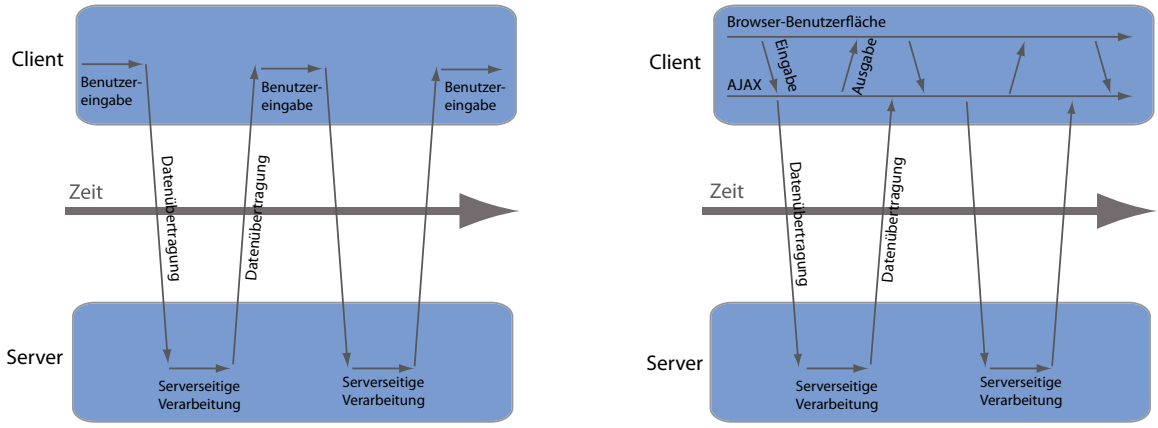


Abbildung 1: synchrone und asynchrone Datenübertragung

Ermöglicht wird die asynchrone Datenübertragung durch ein im Browser integriertes Objekt: Dem XMLHttpRequest-Objekt. Auf dieses Objekt kann mit JavaScript zugegriffen werden um eine asynchrone Anfrage an den Server zu starten oder die Antwort des Servers zu verarbeiten. Dabei werden in der Server-Antwort von AJAX die Datenformate XML und Text unterstützt. Die einzelnen HTML-Elemente der Webseite können dann über das Document Object Model (DOM) mit Hilfe von JavaScript dynamisch an die Server-Antwort angepasst werden. Darüber hinaus können zusätzliche Layout-Änderungen über die Formatregeln in Cascading Style Sheets (CSS) realisiert werden. Im Folgenden soll eine genauere Beschreibung der Basistechnologien helfen, den Zusammenhang zwischen den einzelnen Technologien besser zu verstehen.

2.1.2 Basis-Technologien

2.1.2.1 XMLHttp-Request

Das XMLHttp-Request-Objekt ist zuständig für die asynchrone Datenübertragung zwischen dem Browser und dem Server und wird auch als AJAX-Engine bezeichnet. Das Objekt stellt Methoden und Eigenschaften zur Verfügung, um mittels JavaScript eine Server-Anfrage abzuschicken oder auf die Antwort des Servers zuzugreifen.

2.1.2.2 (X)HTML

Die (eXtensible) Hypertext Markup Language ist eine Auszeichnungssprache zur logischen Beschreibung der Struktur eines Dokuments im Internet und ist somit auch verantwortlich für die Darstellung der Inhalte von AJAX-Anwendungen. Die Seiteninhalte werden durch so genannte„Tags“ ausgezeichnet. Dabei unterscheidet man zwischen logischen Auszeichnungen wie beispielsweise Überschriften, Absätze oder Zitaten und physischen Auszeichnungen wie zum Beispiel Schriftformatierung oder Positionierung. Mit der Einführung von HTML 4 wurde begonnen, diejenigen Elemente und Attribute, die direkt für die Präsentation des Dokuments zuständig waren und keine ausgabe-unabhängige Strukturierung ausdrückten, also die physischen Auszeichnungen, schrittweise aus HTML auszuschließen.
Mit der Einführung von XHTML hat es sich im modernen Webdesign durchgesetzt, die Dokumente konsequent mit CSS zu formatieren. Der strukturierte Inhalt und das jeweilige Layout können dadurch getrennt definiert werden. XHTML Transitional 1.0 ist der letzte Dokumenttyp, welcher noch Layout-Elemente wie etwa *<font>* oder *<b>* enthält. In moderneren Dokumenttypen wie XHTML Strict 1.0 sind nur noch aus Gründen der Abwärtskompatibilität zu den Transitional Dokumenttypen einige wenige Layout-Elemente enthalten. In den Dokumenttypen XHTML Basic oder XHTML 2 sind schließlich gar keine Layout-Elemente mehr vorhanden. Für die physische Gestaltung von XHTML-Elementen soll nur noch auf externe CSS-Regeln verwiesen werden.
Darüber hinaus wurde mit XHTML auch eine strengere Syntax eingeführt:

- alle Tags und Attribute müssen klein geschrieben werden
- alle Attribute müssen in Anführungszeichen gesetzt werden und dürfen nicht alleine stehen
- alle Tags müssen korrekt geschlossen werden (z.B. *
*)

Im Gegensatz zu seinem Vorgänger HTML, welcher mittels SGML (Standard Generalized Markup Language) definiert wurde, verwendet XHTML die strengere und einfacher zu parsende SGML-Teilmenge XML als Sprachgrundlage, welche flexibel um beliebige Attribute und Elementtypen erweitert werden kann[3].

2.1.2.3 DOM

Die ineinander verschachtelten HTML-Elemente der Dokumentstruktur einer Webseite werden als Knoten (engl. ***„nodes“***) bezeichnet. Alle Knoten gemeinsam bilden den so genannten Objektbaum, der im Browser-Speicher aufgebaut wird. Die ***„node“***-Schnittstelle ist eine vom Document Object Model bereitgestellte standardisierte Schnittstelle für den einheitlichen Zugriff auf diese Knoten. Sie liefert Methoden und Attribute um beliebige Knoten über JavaScript anzusteuern, abzufragen oder zu verändern. AJAX-Anwendungen können somit die Antwort des Servers direkt über das DOM in die HTML-Struktur integrieren und einzelne Bereiche der Webseite neu generieren.

3 [WIKIPEDIA01]

2.1.2.4 CSS

Die Formatregeln der Cascading Style Sheets bestimmen wie die einzelnen HTML Elemente dargestellt werden sollen. Mit ihrer Hilfe kann beispielsweise eine genaue Schriftgröße mit bestimmten Zeilenabstand und Laufweite oder die exakte Positionierung von Elementen festgelegt werden. Es gibt drei Möglichkeiten um Cascading Stylesheets in eine Webseite zu integrieren:

- innerhalb der HTML-Elemente (***„inline"***)
- im <header>-Tag des HTML Dokuments (***„embedded"***)
- separat in einer externen CSS-Datei.(***„extern"***)

Die Formatregeln in einer externen Datei auszulagern hat den Vorteil, dass die CSS-Datei für mehrere Webseiten verwendet werden kann. Neben der besseren Wartbarkeit des Codes müssen die Formatanweisungen bei plattformübergreifendem Einsatz nur an die unterschiedlichen Ausgabeformate wie zum Beispiel Mobiltelefone oder Handhelds angepasst werden, da die Struktur und der Inhalt unverändert bleiben.

2.1.2.5 JavaScript

JavaScript ist eine kompakte Scriptsprache zur Erweiterung des HTML-Befehlssatzes, mit der auch ohne ausgereifte Programmierkenntnisse objektorientierte Anwendungen in Internetseiten implementiert werden können. Da sie direkt in das HTML-Dokument eingebunden und lokal von der Rendering-Engine des Browsers interpretiert wird, benötigt man keine zusätzlichen Editoren oder Entwicklungsumgebungen. Die Einbindung kann dabei auf drei unterschiedliche Möglichkeiten erfolgen:

- als Parameterwert direkt in einer HTML-Anweisung (***„inline"***-Referenz)
- als ***<script>***-Block an einer beliebigen Stelle im Dokument (direkte Notation)
- als externe JavaScript-Datei

Eine Auslagerung des Codes ist vor allem dann sinnvoll, wenn bestimmte Funktionalitäten in verschiedenen Webseiten verwendet werden. Auch hinsichtlich der Wartung und Verwaltung des Codes ist eine strikte Trennung von Struktur und Funktionalität empfehlenswert.
JavaScript dient beispielsweise der Verifikation von Formularen, dem Bereitstellen zusätzlicher Informationen bei Verweisen, dem Hinzufügen von Effekten in Inhalten, oder der Berechnung mathematischer Formeln.
Seit JavaScript 1.6 wird das „ECMAScript für XML" (E4X) unterstützt womit sich XML-Strukturen direkt in JavaScript erzeugen lassen. ECMAScript ist der offizielle von der European Computer Manufacturers Association vergebene Name für JavaScript. Allerdings wird „JavaScript für XML" nur von wenigen Browsern unterstützt. Der Internet Explorer bietet keine Unterstützung, während Firefox den Code bei Angabe des JavaScript-MIME Typs ***„text/javascript;e4x=1"*** interpretiert.

2.1.2.6 XML

Die „Extensible Markup Language“ ist eine Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten in Form von Textdateien[4]. Mit XML ist es möglich Informationen so darzustellen, dass sie leicht maschinell weiter verarbeitet werden können. Besonders beim Austausch von Daten zwischen unterschiedlichen Anwendungen und Plattformen eignet sich XML aufgrund seines standardisierten und universellen Formats besonders gut. Auch AJAX-Anwendungen können XML-Daten im HTTP-Response des Servers empfangen und interpretieren.
Der Aufbau eines XML-Dokuments ist recht einfach. Am Anfang jedes Dokuments steht immer die XML-Deklaration mit Angabe der verwendeten XML-Version und Zeichencodierung: ***„<?xml version=“1.0“ encoding=“utf-8“?>“***. Beachtet man die grundlegenden Regeln der Syntax erhält man ein „wohlgeformtes“ XML-Dokument. Um jedoch ein gültiges XML-Dokument zu erstellen muss es die Regeln einer Grammatik einhalten, die entweder in Form einer Document Type Definition (DTD) oder als XML-Schema vorliegt.

2.1.3 AJAX Beispiele

Einer der ersten Versuche AJAX auf einer Webseite zu integrieren startete Google mit der Erweiterung seines konventionellen Suchdienstes um eine interaktive und unterstützende Echtzeitsuche: „Google Suggest“[5] (engl. suggest = vorschlagen). Wird vom Nutzer im Suchfeld ein Begriff eingegeben, werden ihm Suchbegriffe in einem Listenfeld zusammen mit der entsprechenden Trefferanzahl vorgeschlagen, die mit den bisher eingegebenen Zeichen beginnen.
Um diese Funktion realisieren zu können verwendet Google Suggest die asynchrone Datenübertragung. Bei jeder Eingabe eines Buchstabens in das Suchfeld sendet die AJAX-Engine einen HTTP-Request mit dem eingegebenen Buchstaben als Parameter an den Server. Aus einer Datenbank mit Suchbegriffen, die andere Nutzer eingegeben haben werden dann diejenigen Suchbegriffe herausgefiltert, die mit dieser Zeichenkette beginnen. Die nach Trefferanzahl sortierten Suchbegriffe wer den dann an den Browser zurückgesendet und in einer Liste ausgegeben.

Abbildung 2: Google Suggest

4 [WIKIPEDIA02]

5 Erreichbar unter http://labs.google.com/suggest/

Ein weiteres einfaches Beispiel zur Verwendung von AJAX ist der von Google im Februar 2005 gestartete Kartendienst „Google Maps“[6]. Das Ergebnis der Suche nach Orten, Hotels, Restaurants und anderen Objekten wird je nach Wunsch auf einer Landkarte oder einer Satellitenkarte angezeigt. Darüber hinaus können in der zusätzlichen Hybridansicht informative Kartendaten wie Straßennamen und Sehenswürdigkeiten über die Satellitenbilder gelegt werden.
Die interaktive Karte von Google Maps kann vom Nutzer mit der Maus in jede beliebige Richtung verschoben werden. Dazu wurde die Karte in mehrere quadratische Kacheln aufgeteilt. Wird die Karte so weit verschoben, dass eine neue Kachel in den sichtbaren Bereich tritt, wird diese mit AJAX nachgeladen. Bei Verwendung der Zoomfunktion wird eine neue, an die Zoomstufe angepasste Karte via AJAX geladen und über alle sichtbaren Kacheln gelegt. Damit die Kacheln schneller angezeigt werden können, werden einmal herunter geladene Kacheln für die Dauer der Sitzung im Browser zwischen gespeichert[7].

2.2 Flash – Eine Einführung

Flash ist eine Entwicklungsumgebung zur Erstellung multimedialer Inhalte, so genannter „Flash-Filme“. Die Rohdateien werden im Format „.fla“ hinterlegt und werden zu einer Datei mit der Endung „.swf“ (Shockwave Flash) übersetzt, und dabei auf Wunsch auch komprimiert. Diese Datei kann nicht ohne weiteres verändert werden. Die Änderungen müssen in der Rohdatei vorgenommen werden.
Im Jahre 1996 brachte die Softwarefirma FutureWave ein völlig neuartiges Animationsprogramm, mit dem Namen „FutureSplash-Animator“, auf den Markt. Das dazugehörige Browser Plug-In trug den Namen „FutureSplash“.
Noch im selben Jahr wurde FutureWave von Macromedia aufgekauft und im Jahr 1997 wurde aus „FutureSplash-Animator“ „Flash“. Noch im selben Jahr veröffentlichte Macromedia die zweite Flash-Version und einen erweiterten Player. Bis heute brachte Macromedia fast jedes Jahr eine neue Version des Programms auf den Markt, wobei 1999 mit Flash-Version 4 die Programmiersprache „ActionScript“ zum ersten Mal auftauchte. Es folgten sechs weitere Flash-Versionen und zwei weitere ActionScript-Versionen. Mehr zum Thema ActionScript wird im Kapitel Flash-Entwicklung erläutert.

2.2.1 Flash-Entwicklung

2.2.1.1 Flash-Entwicklungsumgebung

Die Wichtigen Bedienelemente für den Flash-Entwickler sind die Timeline, die Bühne und die Bibliothek. Die Flash-Filme sind framesbasiert. Die Sekunde wird in einzelne Bilder eingeteilt. In wie viele entscheidet der Entwickler. Auf der Timeline lässt sich jedes einzelnes Frame, Schlüsselbild genannt, anwählen und manipulieren. Die Bühne, auf der alle Elemente per Drag and Drop platziert werden können, ist der sichtbare Bereich, der in der später kompilierten SWF zu sehen ist. In der Bibliothek werden alle vorhanden Objekttypen angezeigt. Von jedem Objekttyp kann es mehrere Instanzen

6 Erreichbar unter http://maps.google.de/
7 [LUDWIG01]

geben, die durch verschiedene Instanz-Namen voneinander unterschieden werden können.
In Flash gibt es mehrere Ansichtselemente wie zum Beispiel Movie-Clips, Buttons, Grafiken oder Textfelder die auf der Bühne platziert werden können. Diese können dann im zeitlichen Verlauf entweder in der Timeline, oder über ActionScript verändert und animiert werden (siehe dazu 2.2.1.2 ActionScript).

2.2.1.2 ActionScript

Seit Flash-Version 4 besteht die Möglichkeit, die Animationen durch Programmierung zu erweitern. Die dafür entwickelte Programmiersprache heißt ActionScript und die neueste Version ist ActionScript 3. Diese Neuerung vereinfacht die Animation erheblich und ermöglicht erst die Interaktion und Datenverarbeitung. Die meisten Anwendungen in Flash wären ohne ActionScript nicht möglich.
Mit der zweiten Version wurde ActionScript zu einer objektorientierten Programmiersprache, angelehnt an ECMAScript 4. ActionScript 3 ist noch strenger strukturiert und erinnert immer mehr an die Progammiersprachen Java und C++.
Der ActionScript-Programmiercode lässt sich an drei verschiedenen Stellen platzieren: Auf einem Schlüsselbild, auf einem Movie-Clip oder einem anderen Anzeigeelement, oder in einer externen Datei mit dem Suffix „.as". Der Code wird dann ausgeführt wenn das Schlüsselbild erreicht ist, das Anzeigeobjekt erzeugt wird, oder wenn die externe Datei importiert wird.
Es hat sich mit der Zeit eingebürgert dass der Code, bis auf wenige einfache aber essentielle Befehle (wie z.B.: stop()), ausschließlich in externe Dateien geschrieben wird (für große Projekte). Dies hat auch seinen Grund: Dadurch wird das ganze Programm übersichtlicher und es ist einfacher einzelne Funktionalitäten zwischen verschiedenen Programmierern aufzuteilen. Durch sinnvolle Instanz-Namen und die Punktnotation können alle Elemente im Projekt von jeder Stelle aus angesprochen werden. Folglich ist sofort klar, welcher Code zu welchem Objekt gehört. Dabei ist jede Datei eine Klasse und die Ordner in der ersten Ebene sind Pakete (Packeges) in denen die Klassen verwaltet werden.

2.2.2 Integration

2.2.2.1 Umgang mit externen Daten

Flash kann auf Dateien zugreifen, die außerhalb der SWF liegen. Je nach Datei muss dafür eine besondere Klasse oder ein bestimmtes Ladeverfahren benutzt werden. Diese Daten können zum Beispiel Text, Grafiken, Videos, Sounds oder XML-Dateien sein.
Die Klasse „Loader" ist die Standardklasse für den Umgang mit Bildern oder anderen SWFs. Sie wird benutzt um die Datei zu laden, den Ladestatus zu überwachen und um die Datei anzuzeigen. Andere Klassen wie „URLLoader", „NetStream", „Video" oder „Sound" werden für die Verwendung spezieller Datentypen benutzt.
Für den Umgang mit XML bietet Flash auch eine besondere Klasse. Die Klasse „XML" ermöglicht den Entwickler einfach auf die Datenstruktur der Datei zuzugreifen und zu verändern.

2.2.2.2 Einbetten in HTML

Es gibt mehrere Möglichkeiten eine SWF-Datei in HTML einzubetten:
Die zwei ursprünglichen Methoden sind mithilfe eines ActiveX-Control für den Internet Explorer und eines <embed>-Elements für Netscape.
Mit dem <object>-Element wurde eine neue browserübergreifende Methode im Rahmen des HTML4-Standards entwickelt um Flash-Filme darzustellen. Diese Technik funktioniert nur in den neuen Browsern und zeigt im Internet Explorer den Flash-Film erst dann an, wenn er komplett geladen ist. Das widerspricht der wichtigen Streaming-Eigenschaft von Flash (mehr zu den Vor- und Nachteilen von Flash siehe 2.3 AJAX vs. Flash).
Um das Streaming-Problem zu lösen wurden drei neue Ansätze entwickelt. Die Vorgehensweise laut der Technik„Flash Satay" ist die Benutzung von einem zweiten Container-SWF das den eigentlichen Flash-Film lädt. Das zwingt den Entwickler immer zu einem extra Lade-Film und erschwert nur die Parameterübergabe.
Auch mit der Technik„Nested Objekt" wird eine Verschachtelung benutzt, allerdings vom <objekt>-Element. Der Äußere ist nur für den Internet Explorer gedacht und verwendet die alte Technik. Die anderen Browser ignorieren das äußere Element und benutzen nur das Innere nach dem HTML4-Standard, das mit Internet Explorer-Kommentaren für eben diesen Browser versteckt wird.
Das„SWFObjekt" ersetzt, sobald die HTML-Seite geladen ist, einen Platzhalter nachträglich mit dem Flash-Film. Falls kein JavaScript oder Flash aktiviert ist wird der Platzhalter angezeigt.
Im März 2008 ist das verbesserte SWFObject 2 veröffentlicht worden. Diese Methode ist die Aktuellste und Einfachste und wird daher in unserem Projekt verwendet.[8][9]

2.2.3 Flash im Einsatz

Am Anfang seines Daseins war Flash ein reines Animationswerkzeug und meistens als Intro für Webseiten im Internet zu finden. Erst mit ActionScript wurden die Animationen interaktiv.
Heute findet Flash im Web auf unterschiedliche Arten Gebrauch und ist für interaktive Animationen fast alternativlos. Andere offene Möglichkeiten (wie SVG oder SMIL) erfahren bisher keine breite Webbrowser-Unterstützung.

Der klassische Einsatz von Flash im Web ist als Werbebanner oder kleine Animation. Vor allem im Werbebereich oder im E-Commerce wird diese Art der Flash-Filme eingesetzt um die Aufmerksamkeit der Webuser auf das Produkt zu lenken.
Im E-Commerce wird Flash auch für aufwendige Produktpräsentationen oder -konfiguratoren verwendet. (Siehe Abb. 3)

8 [SELFHTML]
9 [ALISTAPART]

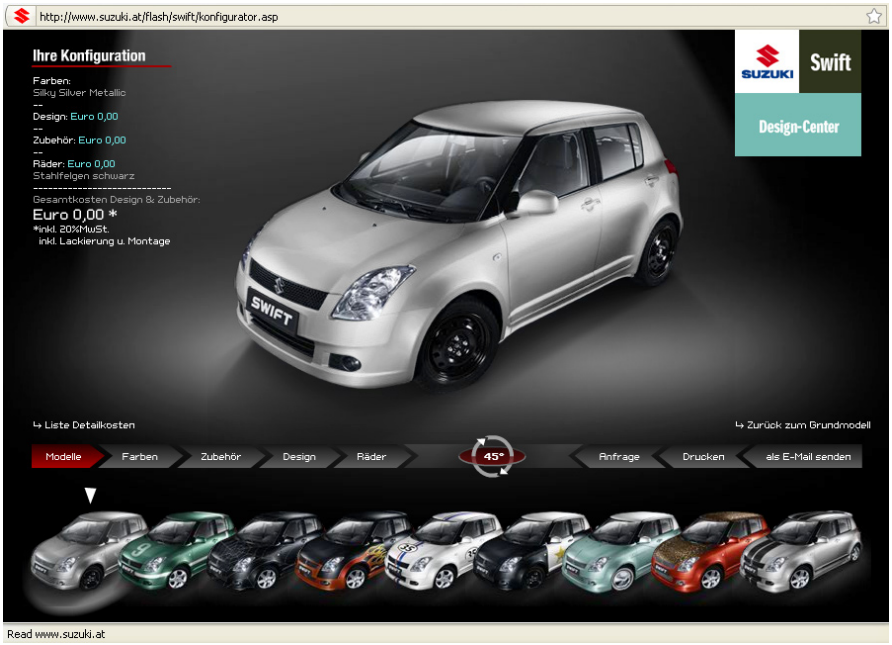


Abb. 3 – Der Suzuki Swift Konfigurator

Des Weiteren können mit Flash auch Trickfilme entstehen, wie zum Beispiel der mittlerweile sehr bekannte Film „Animator vs. Animation“[10]. Da die erstellten Grafiken in Flash Vektorgrafiken sind, wird der Film sehr kompakt und lässt sich dadurch ohne Qualitätsverluste im Internet verbreiten. Auch anspruchsvolle Spiele können mit Flash realisiert werden. Das preisgekrönte Spiel „Get the glass“ [11]hätte aufgrund seiner komplexen Struktur auch eine Desktop-Anwendung sein können, aber mithilfe von Flash lässt sich das Spiel auch sehr gut online spielen (eine schnelle Internetverbindung ist hier natürlich von Vorteil, sonst sind die Ladezeiten schon etwas zu lang). Der Vollständigkeit halber soll hier auch noch ein Beispiel für den Einsatz von Sound in Flash erwähnt werden. „Pepsi move the crowd“ ist eine Anwendung in der der User in Echtzeit spielerisch Sounds verändern kann.[12]

Die gute Komprimierung, die Streaming-Fähigkeit von Flash-Filmen und der dazugehörige Videoformat „FLV“ ermöglichen Videoportale wie „YouTube“ die hochgeladenen Videos schnell und einfach wiederzugeben. (Abb. 4)

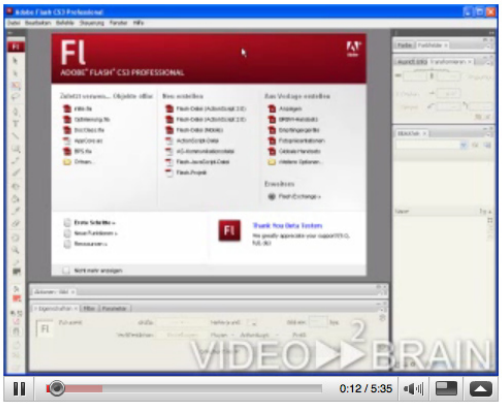


Abb. 4 – YouTube

10 [YOUTUBE]
11 Gewinner der Kategorie „Experience“ auf dem Flash Film Festival Boston 2007
12 Erreichbar unter http://www.pepsidjdivision.com/movethecrowd/

Auch als Teil einer Webseite, wie z.B. als Steuerungsmenü, wird Flash immer mehr verwendet. Auch ganze Webseiten werden teilweise ausschließlich in Flash gebaut.[13] Zu erwähnen ist hier dass bei diesen Seite die künstlerischen Aspekte und nicht die Informationen im Vordergrund stehen.

Um Flash-Dateien betrachten zu können, ist der Flash Player erforderlich, das auch als Webbrowser Plug-In eingebunden werden kann. Seit Januar 2007 ist der Flash Player 9 auch für Linux verfügbar. Die alternativen Open Source-Player „Gnash" und „Swfdec" unterstützen noch nicht alle Funktionen des Flash-Formats.

Vor allem für „Rich Internet Applications" sind Flash-Anwendungen von großer Bedeutung, da die Vorteile von Flash in Sachen Multimedia und Interaktion die Anwendung grafisch und funktional aufwerten. Eine intuitive und desktopähnliche Umgebung kann mit den vielseitigen Möglichkeiten in Flash gut umgesetzt werden. [14]

2.3 AJAX vs Flash

2.3.1. Accessibility

Accessibility wird am besten mit „Zugänglichkeit" oder „Barrierefreiheit" übersetzt. Gemeint sind damit alle Techniken und Strategien, mit denen eine Website für möglichst viele Menschen erreichbar gemacht werden kann. Insbesondere bezieht sich dies auf Menschen mit körperlichen Einschränkungen, wie z. B. Blindheit. [15]

Accessibility in Flash

Um Flash-Anwendungen abspielen zu können, muss sich der Benutzer das Flash-Plug-In für seinen Browser von Adobe herunterladen. Die benötigte Version hängt von dem zu ladenden Flash-Film ab.

Nach einer Studie des Instituts „Milward Brown" besitzt der Flash Player einen sehr hohen Verbreitungsgrad von etwa 99% auf internetfähigen Rechnern in den Hauptmärkten (USA, Kanada, Großbritannien, Deutschland, Frankreich, Japan).

	Flash Player 7	Flash Player 8	Flash Player 9
Mature Markets	99.0%	98.7%	97.7%
US/Canada	99.1%	98.9%	97.8%
Europe	98.5%	97.9%	96.5%
Japan	99.3%	99.3%	98.8%

Tabelle 1 Verbreitungsgrad Flash Player, Millward Brown (Juni 2008)

Der Flash Player erweist große Mängel in der Zusammenarbeit mit Suchmaschinen. Diese haben Probleme mit dem Indizieren von Flash-Inhalten, da der Code für sie unsichtbar ist. Bisher können die Suchanbieter nur statische Textelemente und Links in SWF-Dateien indexieren. Adobe hat deshalb

13 z.B. http://www.music-in-a-bottle.de/
14 [wikipedia03]
15 [GALILEO]

„Google“ und „Yahoo“ eine optimierte Version seiner Flash Player-Technologie zur Verfügung gestellt, um mit Flash generierte dynamische Webinhalte und Rich Internet Applications (RIAs) für Suchmaschinen zugänglich zu machen. „Google“ setzt diese neue Technik bereits ein.[16]

Seit der Version 6 des Flash Players wird die „Microsoft Active Accessibility“ (MSAA) unterstützt. MSAA bietet einen standardisierten Mechanismus zum Austausch von Informationen zwischen Anwendungen und Bildschirmleseprogrammen. Die Accessibility-Klasse muss dazu aber vom Programmierer erst implementiert werden. Außerdem gibt es noch viele weitere Möglichkeiten um Flash-Anwendungen behindertengerechter zu gestalten, wie zum Beispiel das Verknüpfen akustischer Inhalte mit Flash-Objekten, oder das Programmieren von Access-Keys für Benutzer mit motorischen Problemen.
Flash-Anwendungen können nicht nur für das Internet programmiert werden, sondern auch zum Beispiel als eigenständiges Programm für CD-Roms oder für den Desktop. Mit Flash Lite hat man die Möglichkeit interaktive Inhalte für das Mobiltelefon zu erstellen.

Accessibility in AJAX
Um AJAX zu benutzen muss im Browser des Benutzers JavaScript aktiviert werden. Im Internet Explorer muss zusätzlich noch ActiveX aktiviert werden, da dieser sonst den XMLHttp-Request nicht ausführen kann. Ab Version 7 unterstützt auch der Internet Explorer ein natives XMLHttp-Request-Objekt. Etwa 98% der Internetbrowser unterstützen AJAX. Es gibt leider keine genauen Angaben, man kann aber davon ausgehen, dass um die 10% der Internetnutzer JavaScript ausgeschalten haben. Somit ist der Verbreitungsgrad eher kleiner als bei Flash.
Suchmaschinen haben ebenfalls große Probleme mit AJAX. Sie können die Daten, die mit Hilfe von JavaScript nachgeladen werden, nicht finden. Da Suchmaschinen den HTML-Quelltext einer Webseite nach dem Laden prüfen, tragen nachträgliche Änderungen dazu bei, dass die Seite inhaltlich wie auch strukturell modifiziert wird. Die Suchmaschinen erfassen aber trotz der Änderungen immer noch die ursprünglich geladene Seite.
In AJAX kann der Benutzer ohne Maustaste mit der Tabulator-Taste über die Oberfläche navigieren und Kontraste oder die Schriftgröße dynamisch ändern. AJAX-Anwendungen sind nur begrenzt barrierefrei. Ein weitestgehend ungelöstes Problem ist die Darstellung über Screenreader. Diese haben mit JavaScript kein Problem, nachgeladene Inhalte werden aber meist nicht erkannt und Benutzer eines Screenreaders müssen sich deswegen auf ihr eigenes Gefühl verlassen.
Die meisten Browser, wie Nokias OSS Browser, SkyFire und Opera Mobile unterstützen AJAX auf dem mobilen Telefon.

2.3.2 Usability

Beim Thema Usability von Web-Anwendungen geht es nicht um besonders ausgefallene Benutzeroberflächen, sondern um ganz elementare Fragen: Wie gestaltet man eine Anwendung so, dass ein Benutzer rasch und leicht mit ihr arbeiten kann? Wie reduziert man die Zeit, die ein Benutzer benötigt, bis er die Funktionsweise der Anwendung versteht? Wie kann man dafür sorgen, dass die Anwendung effizientes Arbeiten ermöglicht? Hinter diesen scheinbar einfachen Fragen verbirgt sich

16 [ADOBE01]

ein sehr komplexes Problemfeld, mit dem sich ein eigener Forschungsbereich befasst. [17]

Usability in AJAX

Die Abläufe beim surfen durch das World Wide Web sind normalerweise immer gleich: Durch das Anklicken von Hyperlinks lässt es sich durch die verschiedenen Webseiten navigieren. Webseiten mit der AJAX-Technologie haben aber nur wenig mit herkömmlichen Seiten gemein. Deshalb ist der Einsatz von AJAX allein schon ein Usability-Problem.

Bestimmte Funktionen wie die Autovervollständigung oder das Arbeiten per „Drag and Drop" können zwar zur Verbesserung der Usability angesehen werden, stehen aber im Gegensatz zu den Erwartungen der Nutzer. Bei AJAX läuft eine Anfrage ohne das Wissen des Benutzers im Hintergrund ab und dieser erhält keine Rückmeldung, wann dieser Prozess beendet ist. Da beim Laden der Mauszeiger nicht zu einer Sanduhr wird, sollte man bei der Verwendung von AJAX den Ladevorgang mit einem Symbol verdeutlichen, und zwar besonders dann, wenn Daten an einer bestimmten Position auf der Website eingefügt werden, auf den der Benutzer keinen Fokus hat. Es können auch farbliche Hervorhebungen oder Effekte eingesetzt werden, um nachgeladene Inhalte zu markieren.

Ein großer Nachteil bezüglich der Usability ist, dass die Buttons „Vor" und „Zurück" des Browsers nicht genutzt werden können. Außerdem wird das Speichern von Lesezeichen nicht unterstützt. Dies ist auf den dynamisch veränderbaren Inhalt einer AJAX-Webseite zurückzuführen. Es wird nur der statische Zustand, der beim Aufruf der Seite zu sehen ist, im Zwischenspeicher des Browsers gespeichert. [18]

Usability in Flash

In Flash sind praktisch keine Grenzen für den Designer/Programmierer gesetzt. Das ist natürlich ein sehr großer Vorteil, wirkt sich aber oft negativ auf die Usability aus. Flash-Seiten sind oft eine Aneinanderreihung von Filmen und Animationen, die den Benutzer vom eigentlichen Inhalt ablenken. Da es in Flash keine Standartkomponenten gibt, wird die Orientierung für den Betrachter erheblich erschwert. Er muss sich erst auf der Flash-Seite zurechtfinden und die verschiedenen Bausteine seinen gewohnten Komponenten aus HTML-Seiten zuordnen. Das verhindert einen schnellen Zugriff auf bestimmte Inhalte und entfremdet den Zweck der Seite.

Viele standardisierte Vorgänge aus HTML werden nicht unterstützt. So lässt sich der „Vor"- und „Zurück"-Button im Browser nicht benützen, Lesezeichen und „suchen auf der Seite" werden ebenfalls nicht unterstützt. Durch verschiedene Techniken lassen sich aber viele dieser Nachteile weitestgehend beseitigen.

Der große Vorteil von Flash ist die unvergleichbare Möglichkeit der Visualisierung. Komplexe Vorgänge und Navigationen können anschaulich durch vektorbasierte interaktive Animationen präsentiert werden. Durch die Fähigkeit der asynchronen Datenübertragung und des Streamings ist Flash die erste Anlaufstelle für Video oder Sound im Internet.

17 [KAIJÄGER01]

18 [KAIJÄGER01]

2.3.3 Multimedia

Multimedia in AJAX

AJAX bietet dieselben Möglichkeiten wie HTML und CSS zum Einbinden von Multimedia. Die gängigen Formate wie JPG oder GIF und je nach Browserkompatibilität auch PNG können als Bildformate eingesetzt werden. Mathematisch beschriebene Grafiken können mit dem SVG (Scalable Vector Graphics) Format ebenfalls berechnet und zusätzlich über JavaScript animiert werden. Dazu werden einige Open Source Frameworks[19] bereitgestellt.
Schriften können bei AJAX respektive HTML nicht eingebettet werden, weshalb die Auswahl auf die installierten Schriften des Anwendersystems begrenzt sind.
Video und Sound-Dateien können nur über ein zusätzliches Plug-In im Browser abgespielt werden. Am weitesten verbreitet sind dafür der Windows Media Player, QuickTime oder der Real-Player. Die Dateien können zusätzlich über JavaScript gesteuert werden.

Multimedia in Flash

Adobe Flash ist sehr gut für die Wiedergabe von Multimedia-Dateien geeignet. Videos und Sound-Dateien können entweder direkt über die Flash-Entwicklungsumgebung eingebettet, oder als externe Dateien in den Flash-Film nachgeladen werden. Für die Anzeige stehen neben dem progressiven Download, bei dem die Dateien wie jede andere Datei im Internet behandelt wird, auch das weit verbreitete Streaming-Verfahren über einen Streaming-Server zur Verfügung. Als Streaming bezeichnet man aus einem Rechnernetz empfangene und gleichzeitig wiedergegebene Multimedia-Dateien. Des Weiteren können bei der Kodierung des Videomaterials so genannte Cue-Points (Markierungen) gesetzt werden, die dann beim Abspielen des Videos Ereignisse auslösen, Parameter übermitteln oder zur Synchronisierung mit anderen Elementen auf dem Bildschirm dienen. Ab der Version 9 des Flash Players werden neben FLV (Flash Video) auch andere Videoformate, wie der H.264-Codec und HE-AAC, unterstützt. Außerdem kann eine Flash-Anwendung mit der Bestätigung des Benutzers direkt auf Kamera und Mikrofon zugreifen.
Schriften werden bei statischen Texten automatisch in den Flash-Film integriert. Für dynamische Texte lassen sich ganze Schriftarten in den Film einbetten, wodurch der Text auf allen Systemen identisch ist.
Bilder und Grafiken können direkt in der Entwicklungsumgebung als Vektoren erstellt, oder dynamisch von ActionScript generiert werden. Außerdem wird eine große Anzahl von Formaten zum Importieren unterstützt. Die Bilder und Grafiken lassen sich einfach mit ActionScript animieren.

2.3.4.Sicherheit

Sicherheit in Flash

Ein Flash-Film wird auf der Client-Seite ausgeführt und untersagt jeglichen Zugriff auf die Daten der Festplatte des Benutzers. Außerdem kann der Flash Player in den Standarteinstellungen nur auf Dateien aus der Ursprungsdomäne zugreifen, aus der er geladen wurde. Diese Dateien werden in der gleichen Sicherheits-Sandbox (Sicherheitsgruppe) abgelegt und somit als vertrauenswürdig inter-

19 zum Beispiel: http://mootools.net

pretiert. Über domänenübergreifende Richtliniendateien können die Entwickler aber den Zugriff auf weitere Domänen ermöglichen.
Leider wird immer wieder von erheblichen Sicherheitslücken des Flash Players berichtet. Beim „Clickjacking" zum Beispiel kann ein Angreifer mit einem für den Nutzer vermeintlich harmlosen Mausklick unbemerkt Mikrofon und Kamera eines Rechners einschalten und „kapern"[20]. Außerdem werden immer wieder Wege gefunden die Sicherheits-Sanboxes auf weitere Domänen zu erweitern.
Wenn ein Flash-Film veröffentlich wird, wird dieser in Binärcode kompiliert und somit dem Benutzer nicht zugänglich gemacht. Mit diversen Tools[21] kann dieser Zustand aber wieder in eine Projektdatei zurückgeführt werden, weshalb ein Flash-Film keine vertraulichen Informationen beinhalten sollte.

Sicherheit in AJAX

Ajax-Anwendungen benutzen dieselben Mechanismen zur Client-Server-Kommunikation wie traditionelle Web-Anwendungen und haben somit dieselben Sicherheitslücken. Da sie aber oft deutlich komplexer sind, bieten Sie noch mehr Angriffsfläche für potentielle Anschläge. Durch den Umstand, dass der Quellcode offen liegt, und somit Schwachstellen leichter entdeckt werden können, bildet die größte Sicherheitslücke JavaScript. Hier gilt wie bei Flash die „Same-Origin-Policy", also dass das Script nicht auf andere Domänen als die Ursprungsdomäne zugreifen kann. Dies ist aber mit einfachen Mitteln zu umgehen. Durch das Sandboxprinzip des Browsers wird aber prinzipiell gewährleistet, dass nicht auf das lokale Dateisystem des Benutzers zugegriffen werden kann.
Ein erheblicher Anteil an Sicherheitslücken im Internet ist dem so genannten „Cross-Site-Scripting" (XSS) zuzuordnen. Bei dieser Form des „Angriffs" werden Informationen aus einem Kontext, in dem sie nicht vertrauenswürdig sind, in einen anderen Kontext eingefügt, in dem sie dann als vertrauenswürdig eingestuft werden. Aus diesem vertrauenswürdigen Kontext kann dann ein Angriff gestartet werden. In der Regel wird versucht, sensible Daten des Benutzers auszuspähen, um zum Beispiel Benutzerkonten wie E-Mail-Account oder Online-Banking zu übernehmen. Aufgrund der Asynchronität bei AJAX Anwendungen würde solch ein Angriff unbemerkt im Hintergrund ablaufen[22].
Weitere Schwachstellen bildet das so genannte „Cross-Site Request Forgery". Hierbei handelt es sich um einen Angriff auf ein Computersystem, bei dem der Angreifer unberechtigt Daten in einer Webanwendung verändert indem er einen berechtigten Benutzer dieser Webanwendung ausnutzt. Hierzu wird aus dem Webbrowser des Opfers ohne dessen Wissen und Einverständnis ein gefälschter HTTP-Request an die Webanwendung gesendet, so dass bei dessen Aufruf die Webanwendung die vom Angreifer gewünschte Aktion ausgeführt wird.
Auch das ActiveX-Steuerelement „XMLHTTP" des Internet Explorers stellt aufgrund der engen Bindung von ActiveX an Windows-Betriebssysteme ein weiters Risiko dar. Die ActiveX-Controls haben Zugriff auf alle Systemressourcen inklusive der lokal gespeicherten Daten und Programme. Diese Sicherheitslücke kann zur Ausführung von Programmcode missbraucht werden[23].

20 [ADOBE02]
21 zB Sothink SWF Decompiler. Erreichbar unter: http://de.sothink.com/product/flashdecompiler
22 [WIKIPEDIA04]
23 [KAIJÄGER02]

2.4 AJAX und Flash

Bisher stellte sich immer die Frage welche der beiden Technologien besser ist. Ein neuer Ansatz ist es AJAX und Flash zusammen zu verwenden. Wozu dann der Aufwand um zu entscheiden, welches Instrument man einsetzen sollte? Nicht um herauszufinden, ob das Eine besser ist als das Andere, sondern wo es besser ist.

2.4.1 Warum AJAX & Flash?

AJAX und Flash ergänzen sich wunderbar. Durch den Einsatz beider Technologien kann man auch die jeweiligen Vorteile nutzen und nebenbei, die Nachteile vernachlässigen.
Die wichtigste Erkenntnis aus der vorangegangen Recherche ist, dass AJAX und Flash beide ihre spezifischen Schwerpunkte haben, die man dementsprechend in einer Webanwendung einsetzen sollte.
Hat AJAX Nachteile bei Multimedia, kann dies durch Flash wettgemacht werden. Können Texte in Flash-Seiten von Suchmaschinen schwer indiziert werden, so sind AJAX Texte suchmaschinenfreundlicher.

2.4.2 Einsatz

Die Möglichkeit AJAX und Flash gemeinsam für eine Webanwendung zu nutzen besteht schon seit Jahren, doch erst seit etwa 2006 wird dies auch effektiv getan.
Den nennenswerten Anfang machte „Google“ mit seinem Tool „Google Finance“ [24]. Dabei setzt es Flash zur Darstellung eines bestimmten Aktienwertes ein. Außerdem kann man über einen Scrollbalken den ausgewählten Zeitbereich in Flash verändern.
Rechts daneben werden entsprechende Nachrichten zum jeweiligen Aktienkurs und ausgewählten Zeitraum aufgelistet. Dies geschieht über einen AJAX- Request. Verändert sich nun der gewählte Zeitbereich, ändern sich sofort auch die angezeigten Nachrichten.

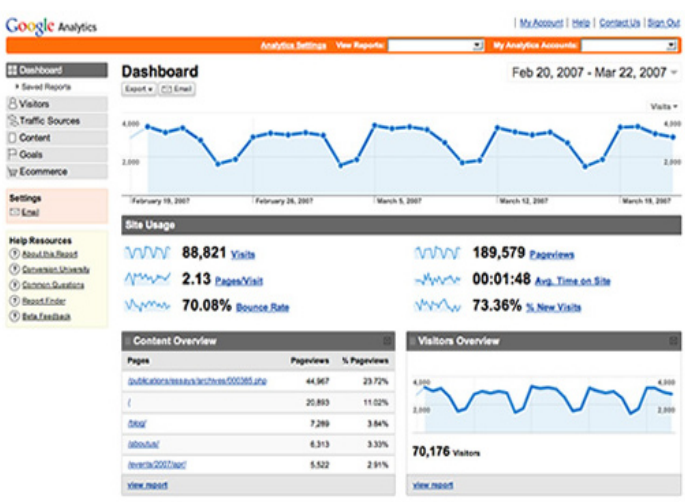


Abbildung 5: Google Analytics

„Google Finance“ ist auch heute noch das Beispiel für hybride Webanwendungen, bei denen sowohl AJAX als auch Flash zum Einsatz kommen. „Google“ hat noch weitere hybride Anwendungen entwickelt, so z.B. „Google Analytics“ [25], ein Dienst, der die Besucher einer bestimmten Webseite erfasst und dem Betreiber detaillierte Informationen über Sie zur Verfügung stellt. Durch seinen enormen Entwicklungsaufwand in neue Technologien ist „Google“ auch weiterhin einer der größten Antreiber in dieser Richtung.

24 Erreichbar unter http://finance.google.com/finance
25 Erreichbar unter http://www.google.com/analytics/de-DE/

Daneben haben auch andere Firmen, wie z.B. „Yahoo!“[26] ähnliche Dienste entwickelt, die AJAX und Flash gemeinsam verwenden.
Weitere Beispiele sind „Empressr“[27] oder „Slideshare“[28]. Während „Empressr“ als PowerPoint- Ersatz gedacht ist um online seine eigenen Präsentation zu erstellen, will „Slideshare“ eine Plattform für Präsentationen jeglicher Art sein, ähnlich wie „YouTube“[29] für Video.
Insgesamt muss man jedoch sagen, dass zwar das Thema brandaktuell ist und auch heiß diskutiert wird, sich scheinbar aber erst sehr wenige konstruktiv an eine Umsetzung gewagt haben. Die Voraussetzungen wären allemal gegeben.

2.4.3 Kommunikation

Auch Adobe hat die Chance erkannt, die eine Kombination beider Technologien bietet und hat in Flash seit Version 8 eine neue Klasse, die „ExternalInterface-Klasse“ fest integriert. Diese Klasse liefert vorgefertigte Methoden zur Kommunikation zwischen AJAX und Flash oder umgekehrt, so dass ganz einfach entsprechende Funktionen der jeweils anderen Seite aufgerufen oder auch Parameter zwischen JavaScript und ActionScript ausgetauscht werden können.
Vor Flash 8 war eine Werteübergabe oder ein Funktionsaufruf nur über die ActionScript Methoden ***„getURL()“*** oder ***„fscommand()“*** sowie die JavaScript Methode ***„SetVariable()“*** möglich. Nachteile waren ganz klar Sicherheitsbeschränkung in der Handhabung von SWFs und die Tatsache, dass nur ein Parameter, meist als String, übergeben werden konnte.
Die ***„ExternalInterface“***-Klasse besitzt zwar dieselbe Funktionalität wie die bisherigen Methoden, ist aber flexibler einsetzbar und kann auch mehrere Parameter übermitteln.

Im Folgenden soll nun nacheinander die Kommunikation von JavaScript nach ActionScript und ebenso die umgekehrte Richtung behandelt werden.

2.4.3.1 JavaScript nach Actionscript

Um die Methoden der ***„ExternalInterface“***-Klasse nutzen zu können, muss man die Klasse zunächst in ActionScript importieren.

```
import flash.external.*;
```

Die Methode ***„addCallback()“*** erfasst eine entsprechende ActionScript Funktion, die dann von JavaScript aus aufgerufen werden kann. Dazu ist nur eine Zeile im ActionScript Code notwendig.

```
ExternalInterface.addCallback(methodName:String, instance:Object,
method:Function);
```

Die 3 Parameter der Funktion ***addCallback()*** im Einzelnen:

methodName: Name der aufrufenden Funktion in JavaScript

26 Erreichbar unter http://de.yahoo.com/
27 Erreichbar unter http://www.empressr.com/
28 Erreichbar unter http://www.slideshare.net/
29 Erreichbar unter http://de.youtube.com/

instance: Objekt, auf das sich this in der Funktion bezieht. Null entspricht root.
method: Name der Funktion, die in Flash ausgeführt werden soll.

Im HTML bzw. JavaScript Code erhält die eingebettete SWF eine eindeutige ID, über die man den Film direkt ansprechen kann. Ein Eventlistener wie etwa „onclick" auf einem Button oder einem sonstigen HTML-Objekt sorgt dann dafür dass die JavaScript- Funktion aufgerufen wird, die mit der ActionScript- Methode verknüpft ist.

<input type="button" onclick="document.getElementById('movie').methodName(param1, param2)" value="JS an AS" />

movie ist die ID der eingebetten SWF
methodName entspricht der aufrufenden JS Funktion

Somit lässt sich von JavaScript aus jede beliebige ActionScript- Methoden aufrufen, wenn diese mit ***addCallback()*** dafür präpariert wurde.

2.4.3.2 ActionScript nach JavaScript

Auch für die umgekehrte Kommunikation gibt es eine Methode in der „ExternalInterface-Klasse". Mit „call()" lassen sich von ActionScript aus JavaScript Funktionen aufrufen und Parameter mit übermitteln.

In ActionScript notiert man dazu folgenden Code:

ExternalInterface.call(methodName:String, [parameter1:Object])

methodName ist der Name der JS Funktion die aufgerufen werden soll, danach folgt eine unbegrenzte Anzahl an zu übergebenden Parametern.

Im JavaScript Code muss im Prinzip nichts verändert werden. Wichtig ist nur, dass es die aufgerufene Funktion gibt.
Ein wichtiger Vorteil der „ExternalInterface-Methoden" ist, dass die Rückgabewerte der aufgerufenen Funktionen sofort zur Verfügung stehen und nicht, wie zuvor notwendig über einen weiteren Funktionsaufruf zurückgesendet werden mussten.

3 Praxisteil

3.1 Projektidee

Unsere Idee ist ein innovativer, interaktiver Messeplan, der auf Webseiten von Messen eingesetzt werden kann. Die meisten Messen bieten bisher nur ihre Messe- und Hallenpläne als PDF an und die Aussteller werden in einer normalen Liste angezeigt. Dadurch kann der Benutzer keinen einfachen Überblick erlangen und nur durch umständliches hin und her Klicken den passenden Stand in einer Halle für einen bestimmten Aussteller finden. In unserem Messeplan werden die Hallen mit den Ausstellern verknüpft und zusätzliche Informationen angeboten. So ist es zum Beispiel möglich ein Imagevideo oder eine Bildergalerie für jeden Aussteller anzuzeigen.
Es kann in Betracht gezogen werden, die Applikation als Messeplaner direkt auf der Messe an Terminals einzusetzen, um den Besuchern ein langes Suchen nach einem bestimmten Aussteller zu ersparen. Außerdem könnte man noch eine mobile Version für Flash und AJAX-fähige Handys entwickeln.

3.2 Projektverlauf

Der Projektverlauf für den Prototyp teilte sich in 4 Abschnitte auf. Im Folgenden sollen diese Teile kurz erläutert werden.

3.2.1 Recherche

Zuerst wurde ausgiebig recherchiert und bereits vorhandene Applikationen, die Flash und AJAX einsetzten, analysiert und getestet. Dabei haben wir herausgefunden, dass dieses Thema schon seit 2006 aktuell ist, die Technik sich aber bis jetzt noch nicht richtig durchsetzen konnte. Die bekanntsten Beispiele, welche weit verbreitet und viel benutzt werden sind die Applikationen von „Google Finance“ und „Yahoo! Finance“ [30].
Dies liegt wahrscheinlich an der Ablehnung der Flash und der AJAX-Programmierer von der jeweiligen anderen Technologie.
Viele Seiten im Internet befassen sich mit dem Vergleich beider Programmiertechniken, nicht aber mit deren Zusammenkunft[31]. Bei diesen Vergleichen geht im Allgemeinen kein Sieger hervor, die Vor- und Nachteile gleichen sich ungefähr aus, und somit liegt es eigentlich nahe beide zu verknüpfen.

3.2.2 Konzeption und Einarbeitung

Für die Konzeption wurden der Aufbau, die Designelemente und alle Funktionen auf der Flash-Seite und auf der AJAX-Seite überlegt. Zusätzlich wurde der Aufbau der Applikation in verschiedene Klassen und Pakete konzipiert. Die gesamte Konzeption ist auf der CD als PDF zu finden.
Gleichzeitig zur Erstellung der Konzeption wurden schon erste Flash-AJAX Beispiele erstellt, um die

30 Erreichbar unter http://finance.google.com/finance

31 [DOT1NE]

Kommunikation zu testen. Da diese Kommunikation aber sehr einfach zu implementieren ist, konnte recht schnell mit dem Projekt begonnen werden.

3.2.3 Implementierung

Flash und AJAX sind zwei unterschiedliche Konzepte, die für unsere Umsetzung zuerst auch als eigenständige Programme implementiert wurden. Auf der Flash-Seite konnte neben den Grundaufgaben wie der Navigation und den verschiedenen Hallen, gleichzeitig die Galerie und der Video-Player entwickelt werden. Auf der AJAX-Seite wurde neben dem Erscheinungsbild und der Kommunikation mit dem Server auch die Administration entwickelt. Die verschiedenen Teile mussten am Ende nur noch zusammengefügt, getestet und verfeinert werden.

3.2.4 Verfeinerung

In der letzten Phase wurde die Applikation getestet und optimiert. Hierzu wurden zusätzlich externe Tester hinzugezogen. Hauptsächlich beliefen sich die Verbesserungen auf Designaspekte und nicht auf die Usability. Die Webseite wurde zum Beispiel noch mal komplett überarbeitet und verschiedene Navigationselemente verständlicher gestaltet. Da es sich bis jetzt noch um einen Prototypen handelt, ist diese Phase noch nicht abgeschlossen. Wir streben an, aus diesem Prototypen ein lauffähiges Programm zu machen, welches dann auch zum Einsatz kommen kann.

3.3 Umsetzung

Im weiteren Verlauf werden nun die verschiedenen Aufgaben von Flash und AJAX aufgezeigt und erklärt.
Es wird darauf hingewiesen, dass die folgenden Codestücke für eine leichtere Verständlichkeit gekürzt und vereinfacht wurden.

3.3.2 Flash

Der Flash-Teil setzt sich aus mehreren Flash-Filmen zusammen. Der Hauptfilm ist immer offen und lädt die benötigten SWFs als Folge einer Benutzeraktion an einen gewünschten Ort. Es wurde weitestgehend versucht objektorientiert mit ActionScript3 zu programmieren. Außerdem wurde der Programmcode der Flash-Filme ausgelagert, um einen direkten Zugriff zu ermöglichen und zu gewährleisten, dass zum Beispiel jede Halle denselben ActionScript-Code benutzen kann.

Die Dateien stehen in folgender Beziehung zueinander:

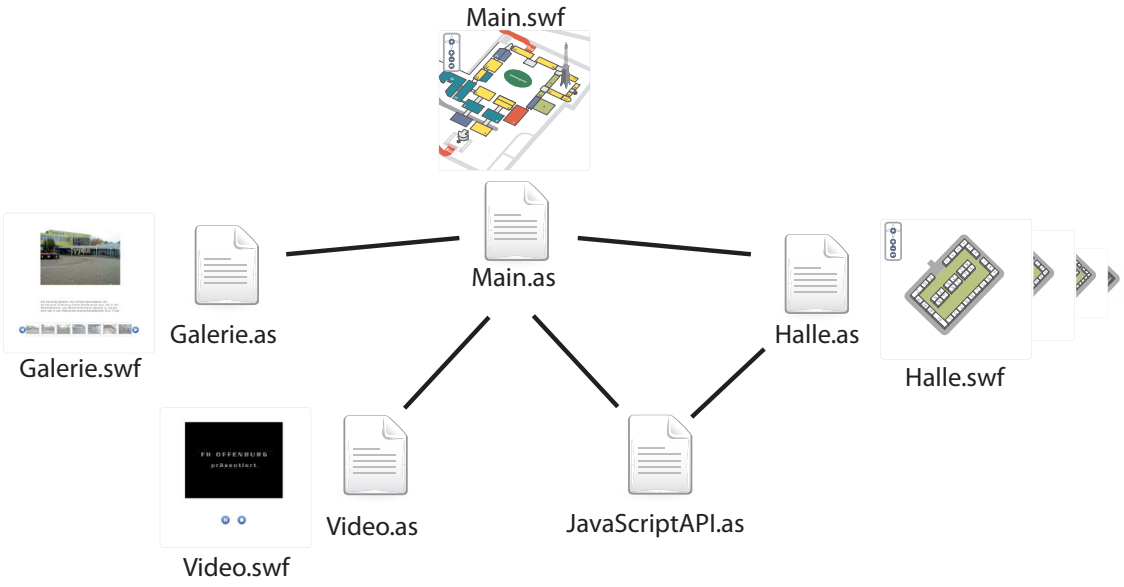
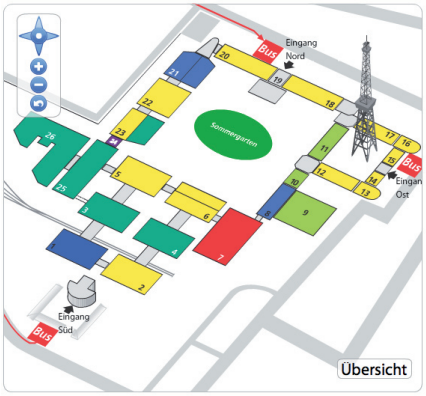


Abbildung 6: Beziehungen der Dateien in Flash

Die AcionScript-Datei *„Main.as“* hat Zugriff auf alle anderen Programmcodes. Sie kann die Flash-Filme *„Galerie.swf“*, *„Video.swf“* und die SWFs der Hallen laden, sowie auf Variablen zugreifen. Außerdem kann von dort über die *„JavaScriptAPI.as“* an JavaScript gesendet werden.
Die ActionScript-Dateien der Hallen können ebenfalls auf die *„JavaScriptAPI.as“* zugreifen und somit auch direkt Daten an JavaScript senden.
Im folgenden Teil werden die einzelnen Flash-Filme näher beschrieben und ihre Beziehungen untereinander dargestellt.

3.3.2.1 MAIN



Der Main-Film ist der Hauptfilm, über den alles gesteuert wird. Er besitzt die ausgelagerte ActionScript-Datei *„Main.as“*. Grafisch befindet sich in der SWF die Übersicht des Messegeländes mit den einzelnen Hallen als Movie-Clips und die Navigation.
Außerdem wird immer der aktuelle Standpunkt rechts unten im Fenster angezeigt.

Abbildung 7: Main.swf

3.3.2.1.1 Navigation

Die Navigation ist so programmiert, dass Sie mit jedem Hintergrund funktioniert, also auch mit allen nachgeladenen Hallen.
Es kann entweder über die grafische Umsetzung navigiert werden oder auch nur mit der Maus oder der Tastatur. Da die Navigation immer auf ein Objekt zugreift, das man dynamisch ändern kann, lässt sich ein Code für die Übersicht und für alle nachgeladen Hallen verwenden.
Die grafische Navigation hat mehrere Buttons. Mit dem ersten Button lässt sich der Hintergrund in alle Seiten verschieben und mit einem Klick wieder an den Ursprungspunkt zurück bewegen.

Abbildung 8: Navigation

Der folgende Code zeigt die einzelnen Schritte die notwendig sind, um eine Verschiebung nach links zu ermöglichen.

```
navi_mc.navi_left.addEventListener(MouseEvent.MOUSE_DOWN, pan_left);
```

Der Movie-Clip mit dem Instanznamen *„navi_left“* wird zuerst mit einem EventListener verknüpft, der die Funktion *„pan_left()“* aufruft, sobald der Benutzer die Maustaste über dem Movie-Clip betätigt.

```
function pan_left( event : MouseEvent ):void
{
    if (target_object.x+target_object.width>233) {
        target_object.x += 8;
    }
}
```

Diese Funktion prüft, ob das *„target_object“* (Hintergrundbild) sich noch an einer Position befindet, in der man es noch nach links verschieben darf und falls dies der Fall ist, wird die x-Koordinate um 8 Pixel erhöht. Nach demselben Prinzip funktioniert das Verschieben in alle Richtungen.

Die nächsten zwei Buttons sind das Hinein-Zoomen und das Heraus-Zoomen. Diese Buttons werden auch mit einem EventListener verknüpft und rufen bei einem Klick entweder die Funktion *„zoomIn()“* oder *„zoomOut()“* auf. Außerdem wird diese Funktion auch über JavaScript für das Scroll-Rad der Maus ausgelöst (3.3.3.4 Kommunikation).
Folgender Code zeigt die Schritte, für das Hinein-Zoomen.

```
function navi_zoomIn( event : MouseEvent ):void
{
    zoomIn( 6/5, rahmen.width/2, rahmen.height/2, target_object );
}
```

Die Funktion *„navi_zoomIn()"* ruft die Funktion *„zoomIn()"* auf und übergibt folgende Parameter. Zuerst den Wert, mit dem gezoomt werden soll. Es folgen eine x- und eine y-Position, die beschreiben, wo gezoomt werden soll. Die wird hier berechnet durch die Breite und die Höhe des Rahmens geteilt durch 2. Als letztes wird das Objekt übergeben, das vergrößert werden soll.

```
public function zoomIn ( scale : Number, originX : Number, originY : Number, mc:Object ):void
{
    affineTransform = mc.transform.matrix;
    affineTransform.translate( -originX, -originY );
    affineTransform.scale( scale, scale );
    affineTransform.translate( originX, originY );

    if (affineTransform.a<3.8 && affineTransform.a>1.4)
          {
          mc.transform.matrix = affineTransform;
          }
}
```

Zuerst wird die Transformationsmatrix des Objektes ermittelt. Die Matrix-Klasse stellt eine Transformationsmatrix dar, die festlegt, wie Punkte eines Koordinatenraums einem anderen Koordinatenraum zugeordnet sind. Das Objekt wird dann relativ zum Ursprung verschoben und danach skaliert. Daraufhin wird es wieder in die Originalposition gebracht. Über eine if-Abfrage wird geprüft, ob sich das Objekt noch in einem festen skalierbaren Bereich befindet, so dass nicht bis ins unendliche skaliert werden kann. Danach wird die neue Transformation auf das Objekt angewandt.

Der letzte Button in der Navigation ist ein „Zurück"-Button, der nur aktiviert wird, wenn eine Halle geladen wurde. Dieser entlädt dann diese SWF wieder (siehe 3.3.2.1.2 SWF laden und entladen).

Eine weitere wichtige Funktion für die Navigation ist das Verschieben per Drag and Drop. Der Benutzer kann dadurch einfach den Hintergrund per Maus hin und her bewegen.
Mit dem Movie-Clip werden zwei EventListeners verknüpft. Einmal für das Drücken der Maustaste und einmal für das Loslassen. Beim Drücken wird die Funktion *„onMouseDown()"* aufgerufen.

```
function onMouseDown( event : MouseEvent ):void
{
var dragBoundsRect : Rectangle = new Rectangle(rahmen.width/2, rahmen.height/2,
-target_object.width, -target_object.height);
target_object.startDrag( false, dragBoundsRect);
intervalId = setTimeout(myDelayedFunction, 500);
}
```

Zunächst muss ein Rechteck erstellt werden, indem sich die linke, obere Seite des gedraggedten Mo-

vie-Clips bewegen darf, sodass der Benutzer den Hintergrund nicht aus dem Flash-Film herausziehen kann. Dieses Rechteck wird jedes Mal neu erstellt, da es sich je nach Zoom-Stufe des Hintergrundes verändert muss. Das folgende Bild zeigt es für die Übersicht in normaler Größe auf.

Das Rechteck wird aus der Höhe und Breite des Movie-Clips, sowie der Mitte des Flash-Filmes (*rahmen.width/2, rahmen.height/2*) berechnet. Danach wird die Methode *„startDrag()"* aufgerufen und dabei das Rechteck mit übergeben. Der Movie-Clip bleibt so lange bewegbar, bis die Freigabe durch einen Aufruf der *„stopDrag()"*-Methode aufgehoben wird.
Um beim Versuch über einem Button zu draggen, diesen nicht auszulösen, wird eine Methode nach einem Timeout von 500 Millisekunden aufgerufen, die überprüft, ob die Maustaste auf einem Button (z.B. um in eine Halle zu gelangen) gedrückt wurde. Falls das der Fall ist, wird eine Variable (*„dragged"*) in der Funktion *„myDelayedFunction()"* auf *„false"* gesetzt, die verhindert, dass der zugehörige Code ausgeführt bzw. die Halle geöffnet wird.

```
function onMouseUp( event : MouseEvent ):void
{
    target_object.stopDrag();
    clearTimeout(intervalId);
    dragged = true;
}
```

Wird innerhalb der 500 Millisekunden die Maustaste wieder losgelassen, vergleichbar mit einem normalen Klick, wird der Timeout abgebrochend und die *„stopDrag()"*-Methode aufgerufen und die Variable wieder auf *„true"* gesetzt, sodass der Code für den Button ausgeführt bzw. die Halle geöffnet werden kann.

3.3.2.1.2 SWF laden und entladen

Jede SWF wird nach demselben Prinzip geladen. Im Folgenden wird beschrieben, wie eine Halle in den Haupt-Film integriert wird. Für jede Halle existiert ein Movie-Clip mit Instanznamen (zum Beispiel *„halle_9"*), die alle in dem Übergeordneten Movie-Clip *„uebersicht_mc"* liegen.

```
for (var a:int = 1; a <= anz_hallen; a++)
{
    uebersicht_mc[„halle_"+[a]].addEventListener(MouseEvent.MOUSE_UP, load_halle_function);
    uebersicht_mc[„halle_"+[a]].id =a;
}
```

Zuerst wird über eine for-Schleife jedem Movie-Clip ein EventListener hinzugefügt. Dieser öffnet immer die Funktion *„load_halle_function"*, wenn auf einen der Movie-Clips geklickt wurde. Außerdem wird jeder Halle eine ID gegeben, um sie später besser erkennen zu können.

```
function load_halle_function(event:MouseEvent):void
{
    var halleNummer:String = event.target.id;
    loadHalle(str_name);
}
```

Bei einem Aufruf wird immer ein MouseEvent-Objekt mit übergeben, welches in diesem Fall der Movie-Clip ist, auf den geklickt wurde (z.B. Halle 9). Über die zuvor vergebene ID lässt sich die Nummer der angeklickten Halle ermitteln, in eine String-Variable speichern und an die Funktion ***„loadHalle()"*** übergeben.

```
public function loadHalle(hNummer)
{
    if (dragged==false )
    {
        var loader:Loader = new Loader();
        loader.unload();
        loader.load(new URLRequest(„swf/hallen/halle_"+hNummer+".swf"));
        loader.contentLoaderInfo.addEventListener(Event.COMPLETE, completeHandler);
    }
}
```

In der Funktion ***„loadHalle()"*** wird zuerst überprüft, ob die Variable ***„dragged"*** gleich ***„false"*** ist. Ist das nicht der Fall wird gerade gedragged und der Code wird nicht ausgeführt.
Falls nicht gedragged wird, wird zuerst ein neues Loader-Objekt mit dem Namen ***„loader"*** erstellt. Durch den Aufruf ***„loader.unload()"*** wird ein mögliches untergeordnetes Objekt dieses Loader-Objekts, das mit der Methode ***„load()"*** geladen wurde, entfernt.
Danach wird die Halle über einen URL-Request in das Loader-Objekt geladen. Die Variable ***„hNummer"*** (Nummer der Halle) dient dabei zum Laden der richtigen Halle.
Mit dem Loader wird über ein ***„contentLoaderInfo-Objekt"***, welches Informationen über den Ladefortschritt beinhaltet, ein EventListener verknüpft, der die Funktion ***„completeHandler()"*** öffnet, sobald die Halle fertig geladen wurde.

```
public function completeHandler(loaderObject:Event)
{
    load_halle = loaderObject.currentTarget.content;
    loadMovie_halle.addChild(load_halle) ;
    load_halle.addEventListener( MouseEvent.MOUSE_DOWN, onMouseDown );
    load_halle.addEventListener( MouseEvent.MOUSE_UP, onMouseUp );
    navi_mc.navi_back.addEventListener(MouseEvent.MOUSE_UP, halle_entladen);
    target_object =load_halle;
}
```

Über den Pfad *„loaderObject.currentTarget.content“* kann man direkt auf die geladene SWF zugreifen und dieses dann über die Methode *„addChild()“* einem sich auf der Bühne befindenden leeren Movie-Clip mit dem Namen *„loadMovie_halle“* hinzufügen.
Dieser SWF werden danach noch die zwei EventListener für das Drag and Drop hinzugefügt, und zum Entladen wird der „Zurück“-Button aktiviert. Damit die Navigation weiß, welches Objekt es z.B. skalieren soll, wird die geladene SWF in der Variable *„target_objekt“* gespeichert.

```
function halle_entladen()
{
    loadMovie_halle.removeChild(load_halle);
    navi_mc.navi_back.removeEventListener(MouseEvent.MOUSE_UP, navi_back_mouse);
    target_object = uebersicht_mc;
}
```

Falls die Halle wieder entladen werden soll, wird die SWF einfach über die Methode *„removeChild()“* wieder aus dem Movie-Clip *„loadMovie_halle“* gelöscht. Außerdem wird der EventListener auf dem „Zurück“-Button wieder entfernt und die Variable *„target_object“* wieder verändert.

Nach dem gleichen Prinzip werden auch der Galerie- und der Video-Film geladen und entladen.

3.3.2.1.3 search()

Die Funktion *„search()“* ist ein wichtiger Bestandteil der Applikation für den Austausch mit JavaScript. Über diese Funktion wird überprüft, welche Hallen bzw. welche Stände in einer Halle gerade in der Flash-Applikation zu sehen sind. Sie wird bei jedem Ereignis (zum Beispiel *„zoomIn()“*) aufgerufen, die eine Änderung der Übersicht, oder der geladenen SWFs einer Halle mit sich gebracht hat. Dazu wird ein Movie-Clip (*„hitTest_mc“*) auf der Bühne platziert und über *„hitTest()“* verglichen, welche Hallen bzw. Stände sich auf diesem Movie-Clip befinden. Folgender Auszug beschreibt diesen Vorgang für die Stände einer Halle.

```
public function search() {
...
    var nummerarray:Array = new Array();

    for (var c:int = 1; c <=anz_staende; c++) {
        if (target_object[„n_“+[c]].clicked==1)
        {
                stand_klicked = true;
                c=anz_staende;
        }
        else {
                stand_klicked = false;
        }
    }
```

Zuerst wird ein Array für die Nummern erstellt, welches am Ende alle Nummern die übertragen werden müssen beinhaltet. Zuerst muss aber über eine for-Schleife überprüft werden, ob einer der Stände in der Halle die Eigenschaft ***„clicked"*** auf 1 stehen hat, denn wenn das der Fall ist, wurde ein Stand angeklickt und es werden die Details dieses Standes angezeigt. Deshalb darf dann kein Array an JavaScript gesendet werden.

```
    if (stand_klicked==false) {
        nummerarray[0]=aktuelle_halle.id;
        for (c= 1; c <=anz_staende; c++) {
            if (target_object[„n_"+[c]].hitTestObject(hitTest_mc)) {
                nummerarray[c]=target_object[„n_"+[c]].id;
            } else {  }
        }
        for (b=1; b<=anz_staende; b++) {
            if (nummerarray_alt[b]==nummerarray[b]) {
            } else {
                b=anz_staende2;
        jsCaller.callJS(nummerarray);
            }
        }
    } else {        }
        nummerarray_alt =nummerarray;
}
```

Wenn kein Stand angeklickt ist, wird in das Array an die erste Stelle die Nummer der Halle geschrieben. Danach wird über eine for-Schleife überprüft, welche der Stände sich auf dem Movie-Clip ***„hitTest_mc"*** befinden. Die IDs dieser Stände werden dann nacheinander in das Array geschrieben. Damit nicht unnötig nachgeladen werden muss, wird das Array noch mit dem vorherigen Array verglichen und falls diese nicht genau gleich sind, wird es an JavaScript gesendet. (Sendevorgang siehe 3.3.2.3 Kommunikation). Als letzter Schritt wird für die nächste Prüfung das alte Array mit dem Neuen ersetzt.

3.3.2.2 Halle

Es gibt für jede Halle einen einzelnen Flash-Film, alle greifen aber auf eine ActionScript-Datei zu. Durch diese dynamische Programmierung ist ein schnelles Verändern der Dateien möglich. Die einzelnen Flash-Filme unterscheiden sich neben den Grafiken nur in zwei Variablen, die die Anzahl der Stände und die Nummer der Hallen beinhalten. Diese werden dann beim Kompilieren jeweils von der ActionScript-Datei gelesen.
Die Aufgabe der ActionScript-Datei ***„halle.as"*** ist es, die Stände nach einem Mausklick einzufärben und die Nummer des ausgewählten Standes, sowie die Hallennummer an JavaScript zu senden.

```
public function stand_waehlen(stand:String) {
    if(halle_load[„n_“+stand].clicked==“0“){
        for (var a:int=1; a<=anz_staende; a++) {
halle_load[„s_“+[a]].transform.colorTransform = new ColorTransform(1, 1, 1, 1, 1, 0, 1, 0);


halle_load[„n_“+[a]].clicked=0;
    }
    var rOffset:Number = transform.colorTransform.redOffset + 100;
    var bOffset:Number = transform.colorTransform.redOffset - 100;
    halle_load[„s_“+[halle_load[„n_“+stand].id]].transform.colorTransform = new ColorTransform
    (1, 1, 1, 1, rOffset, 0, bOffset, 0);
    halle_load[„n_“+stand].clicked=1; jsCaller2.callJS_detail(halle_name,
    halle_load[„n_“+stand].id);
}
```

Nach einem Klick auf einen Stand wird die Funktion ***„stand_waehlen()“*** aufgerufen und die Nummer des Standes ***(„stand:String“***) mit übergeben. Danach wird für diesen Stand über die Eigenschaft ***„clicked“*** überprüft, ob er schon ausgewählt wurde. Wenn das nicht der Fall ist, werden alle Stände über eine for-Schleife und der ***„colorTransform-Klasse“*** der Transform-Klasse auf die Anfangsfarbe gesetzt. Außerdem wird die Eigenschaft ***„clicked“*** der Stande auf 0 gesetzt, sodass nun kein Stand ausgewählt ist. Danach wird der Stand von dem die Nummer übergeben wurde eingefärbt und dessen Eigenschaft ***„clicked“*** auf 1 gesetzt. Durch diesen Vorgang ist immer gewährleistet, dass nur ein einziger Stand angeklickt werden kann. Dieser Stand wird nun an JavaScript gesendet, sodass die Details dieses Standes abgerufen werden können (siehe 3.3.2.3 Kommunikation).
Der nächste Code beschreibt den Ablauf für einen Stand, der schon angeklickt wurde.

```
        …
    else{
        for (var b:int=1; b<=anz_staende; b++) {
        halle_load[„s_“+[b]].transform.colorTransform = new ColorTransform(1, 1, 1, 1, 1, 0, 1, 0);

halle_load[„n_“+[b]].clicked=0;
}
    }
}
```

Falls die Eigenschaft ***„clicked“*** für einen Stand bereits 1 ist, werden alle Stände wieder auf die Anfangsfarbe zurückgesetzt.

3.3.2.3 Kommunikation

Die ActionScript-Datei *„JavaScriptAPI.as“* dient als Schnittstelle für die Kommunikation von Flash zu JavaScript. Da Sie nicht mit einer SWF verknüpft ist, kann man von überall darauf zugreifen. Folgender Code beschreibt diesen Vorgang.

halle.as:
```
var jsCaller : script.JavaScriptAPI;
jsCaller = new JavaScriptAPI();
jsCaller.callJS_detail(halle_name,halle_load[„n_“+stand].id);
```

Zuerst muss eine Variable vom Typ *„script.JavaScriptAPI“* deklariert werden. *„Script.JavaScriptAPI“* ist der Pfad der Klasse *„JavaScriptAPI“* die in dem Paket *„script“* zu finden ist. Danach wird mit dem Operator *„new“* eine neue Klasseninstanz erstellt. Über diese Klasseninstanz kann danach die Funktion *„callJS_detail()“* mit beliebigen Parametern (hier die Hallennummer und die Standnummer) aufgerufen werden.

JavaScriptAPI.as:
```
public function callJS_detail(h_num:Number, s_num:Number)
{
    var jsResponse : Object;
    jsResponse = ExternalInterface.call(„rcvFlash_detail“, h_num,s_num);
}
```

Diese Parameter werden an die ActionScript-Datei *„JavaScriptAPI.as“* übergeben. Dort muss zuerst ein Objekt erstellt werden, über das der Sendevorgang an JavaScript aufgerufen wird.
Bei der *„ExternalInterface-Klasse“* handelt es sich um eine Anwendungsschnittstelle, die eine Kommunikation zwischen ActionScript und JavaScript ermöglicht. Die Methode *„call()“* ruft die bereitgestellte Funktion *„rcvFlash_detail()“* auf und übergibt die Argumente *„h_num“* und *„s_num“*, hier die Hallennummer und die Standnummer.

Nach dem gleichen Prinzip funktioniert auch die Übergabe des Arrays in Abschnitt 3.3.2.1.3 search().

Von JavaScript werden ebenfalls Daten an den Flash-Film gesendet. Die genaue Beschreibung des Sendevorganges in JavaScript wird in 3.3.3.4 Kommunikation beschrieben.
Der folgende Code beschreibt den Vorgang auf Flash-Seite am Beispiel der Zoom-Funktion beschreiben.

```
ExternalInterface.addCallback(„setZoom“, ZoomIt);
...
function ZoomIt( delta_JS ):void
{
```

```
    if ( delta_JS > 0 ) {
        zoomIn(6/5, rahmen.width/2, rahmen.height/2, target_object);
    } else {
        zoomOut( 5/6, rahmen.width/2, rahmen.height/2, target_object);
    }
}
```

Über die Anwendungsschnittstelle *„ExternalInterface“* und der Methode *„addCallback()“* werden die Daten aus JavaScript erhalten. Nach einem erfolgreichen Aufruf von *„addCallback()“* kann die registrierte Funktion im Flash Player aus JavaScript- oder ActiveX-Programmcode im Container aufgerufen werden. Der erste Parameter beschreibt den Namen, mit dem die ActionScript-Funktion von JavaScript aus aufgerufen werden kann, der zweite, die Funktion an die alle Parameter in ActionScript gesendet werden sollen.
In diesem Beispiel wird in JavaScript die Funktion *„setZoom()“* aufgerufen und der Parameter *„delta“* des Mausrades mitgeschickt. Dieser Parameter wird an die Funktion *„zoomIt()“* übergeben und dort nach einem positiven oder negative Wert überprüft. Je nach dem wird die entsprechende Funktion zum herein oder heraus zoomen aufgerufen.
Nach demselben Prinzip wird auch das Laden der Galerien und der Videos, das Öffnen einer Halle oder das Markieren eines Standes aus JavaScript gewährleistet.

3.3.2.4 Galerie

Um unser Projekt multimedialer zu machen, wurden eine Bildergalerie und ein Videoplayer erstellt (zu Video siehe 3.3.2.5). Bevor die Bildergalerie erklärt wird soll zuerst die Ordnerstruktur für die Multimedia-Dateien erläutert werden.

Neben den Ordnern für z.B. JavaScript, SWF, CSS und PHP gibt es auch einen Ordner für die Galerie und das Video. In diesem Ordner wird beim Erstellen der Hallen für jede Halle einen Ordner erzeugt. Diese Hallen-Ordner beinhalteten wieder für jeden Stand darin je einen Ordner.
Somit hat jeder Stand seinen eigenen Ordner. Hier werden alle Dateien gespeichert die für die Galerie wichtig sind:
eine XML, welche die Datenstruktur für die Galerie angibt. Hier stehen alle Dateinamen der Bilder und die dazugehörigen beschreibenden Texte. (siehe Abb.2)
Die eigentlichen Bilder.
Wieder einen Unterordner der die Bilder in einer kleinen Version (genannt Thumbnail) beinhaltetet.

Abbildung 9 : beispielhafte Ordnerstruktur

```
<?xml version="1.0" encoding="utf-8"?>
 <system>
   <bild href="1.jpg" info="Beschreibender Text für Bild 1" />
   <bild href="2.jpg" info="Beschreibender Text für Bild 2" />
   ...
 </system>
```

Aufbau der XML-Datei für die Galerie

Wenn im AJAX-Teil auf den Button für die Bildergalerie geklickt wurde, wird eine SWF mit dem Namen ***„galerie.swf"*** aufgerufen. Diese SWF befindet sich im gleichen Ordner wie die ***„main.swf"*** und existiert nur einmal. Das heißt, egal welcher Stand angeklickt wurde, es wird immer dieselbe SWF geladen.
Die ***„galerie.swf"*** erhält nämlich von der ***„main.swf"*** als Parameter die Hallen- und Standnummer. So kann dann der Pfad zum richtigen Standordner ermittelt und somit auch die richtige XML geladen werden. Anhand dieser XML wird das Aussehen des Flash-Teils dynamisch generiert und die Thumbnails, das große Bild, sowie der passende Text angezeigt.

3.3.2.5 Videoplayer

Das Abspielen des Videos erfolgt auf ähnliche Weise wie das Anzeigen der Bilder in der Bildergalerie.
In dem Standordner befindet sich neben den Galerie-Dateien auch optional ein Video im ***„flv"***-Format. Die Spezifikationen unseres Projektes erlauben pro Aussteller nur ein Video (oder keins). Wenn es vorhanden ist erscheint im AJAX-Teil ein Button zum Videoplayer. Wird dieser angeklickt, wird wie auch hier im Flash-Teil immer die gleiche SWF ***„video.swf"*** in die ***„main.swf"*** geladen und die zwei Parameter Halle- und Standnummer übergeben.
Im dem Fall wird keine XML benötigt da nur eine, oder keine, Datei vorhanden ist und eine Strukturierung der Daten per XML überflüssig ist. Der passende Pfad wird wieder ermittelt und die richtige FLV wird aus dem richtigen Ordner vom Server geladen und abgespielt.

3.3.3 AJAX

Der zweite große Bereich unseres Projektes besteht aus einer vernünftigen Darstellung der Stände und Hallen in HTML. Je nach ausgewählter Halle und den derzeit sichtbaren Ständen werden andere Informationen gezeigt.
Sich dynamisch ändernde Inhalte könnte man über eine dynamische Skriptsprache, wie z.B. PHP oder ColdFusion, realisieren. Dabei würden aber auch andere Teile der Webseite unter anderem auch der Flash-Film neu geladen werden. Dies hätte zur Folge, dass die bisher getätigten Benutzeraktionen in der SWF verloren gingen und man von neuem den entsprechenden Stand auswählen müsste.

Außerdem wird bei dieser Variante eine vergleichsweise große Menge an Daten übermittelt, da immer die ganze Seite neu geladen wird.
AJAX bietet dagegen den Vorteil nur die relevanten Daten zu aktualisieren, die übertragene Datenmenge reduziert sich und die Interaktionen des Benutzers in der SWF bleiben erhalten.

3.3.3.1 XMLHttp-Request

Wie schon im Theorieteil erwähnt, sendet AJAX mittels des XMLHttp-Request Objektes über JavaScript eine HTTP-Anfrage an den Server, welcher daraufhin die entsprechende Antwort, also die gewünschten Informationen zurücksendet.
In unserem Projekt sind alle Informationen zu den Hallen, sowie zu den Ständen in verschiedenen XML- Dateien auf dem Server hinterlegt.

Abbildung 10: XML-Dateien

Der Aufbau, der ***„halle9.xml"*** beispielsweise sieht folgendermaßen aus:

```
<?xml version="1.0" encoding="utf-8"?>
<system>
<aussteller id="9.1" name="Atlus U.S.A., Inc." halle="9" nr="1">
<kategorie>
<hauptkategorie name="Software"/>
<unterkategorie name="PC Spiele"/>
</kategorie>
<firma>
<adresse strasse="199 Technology Drive, Suite 105" plz="CA 92618" ort="Irvine" land="USA"/>
<kontakt telefon="" fax="" mail="" web="http://www.atlus.com/"/>
</firma>
<beschreibung>Beschreibungstext</beschreibung>
<media video="false" picture="false"/>
</aussteller>
<aussteller id="9.2" name="Pipeworks" halle="9" nr="2">
...
```

Jeder Aussteller hat eine eindeutige ID, die sich aus der Hallen- und der Standnummer zusammensetzt, sowie einen Namen.

Darunter gibt es den Kindknoten Kategorie mit Haupt- und Unterkategorie, den Kindknoten Firma, mit Adress- und Kontaktangaben, den Kindknoten Beschreibung sowie den Kindknoten Media, der angibt, ob eine Bildergalerie existiert und ein Video hinterlegt ist.
Die XML für alle Hallen (*„overview.xml“*) ist ähnlich aufgebaut, allerdings gibt es dort nur den Kindknoten Kategorie.

Je nachdem welche Ansicht man gerade im Flash-Teil sieht, werden verschiedene XMLs angefordert:

```
if(halle == 0) {
    var url = „xml/overview.xml“;        // Übersichts XML
}
else {
    var url = ‚xml/halle‘+halle+‘.xml“;  // XML einer bestimmten Halle
}
request.open(‚post‘, url, true);         // Request öffnen
request.send;                            // Request senden
```

Die empfangene Antwort, in unserem Fall also die XML- Datei, wird nun von JavaScript weiterverarbeitet.

```
var xmldoc = request.responseXML;
root_node = xmldoc.getElementsByTagName(‚system‘)[0];
showResponse();
```

Sobald die Datei in ein *„XMLDocument-Objekt“* umgewandelt wurde, wird die Funktion *„showResponse()“* aufgerufen.

3.3.3.2 Darstellung der Inhalte

Die Funktion *„showResponse()“* wählt aus, welche Ansicht erzeugt werden soll, je nachdem ob es eine Übersicht über mehrere Stände gibt oder ein einzelner Stand ausgewählt wurde und dazu Detailinformationen angezeigt werden.

```
function showResponse() {
    if (detail != -1) {
        include_js(‚js/detail.js‘);          // Detailansicht
    }
    else {
        include_js(‚js/overview.js‘);        // Übersicht
    }
}
```

Im Folgenden wird nun die Darstellung mittels der „overview.js“ genauer thematisiert. Interessant zu erwähnen ist, dass diese Datei zur Generierung sowohl der Übersicht über alle Hallen, als auch über alle Stände einer Halle verwendet wird. Möglich wird dies, durch die ähnliche XML- Struktur der jeweiligen Dateien.

Im oberen Bereich der AJAX- Seite befindet sich eine Navigation, die das Blättern, durch die einzelnen Stände oder auch die Übersicht ermöglicht. Dadurch wird vermieden, alle sichtbaren Stände untereinander anzuzeigen, was zu einer endlosen Darstellung führen würde. Außerdem kann man wieder in die übergeordnete Ansicht wechseln. Eine weitere Möglichkeit wäre mittels JavaScript zu scrollen1, dies ist eine Variante, die man evtl. in Zukunft noch einbauen könnte.

Zur Darstellung der einzelnen Hallen bzw. Stände in der Übersicht, werden zunächst die entsprechenden Container mit JavaScript erstellt.

```
var reihen = „“;
for (v=anfang;v<flash_array.length&&v<akt_seite*max_anzahl;v++) {
    nr = flash_array[v];
    reihen = reihen.concat(,<div class=“aussteller“ id=“aussteller‘ + nr + ,“>\n<\/div>\n‘);
}

document.getElementById(„ajax“).innerHTML = reihen;
```

Die Container erhalten eine eindeutige ID *„aussteller1“*, *„aussteller2“*, …, damit man ihnen später weitere Inhalte zuordnen kann. Über den Methodenaufruf *„innerHTML“* werden schließlich die * *-Container erstellt.

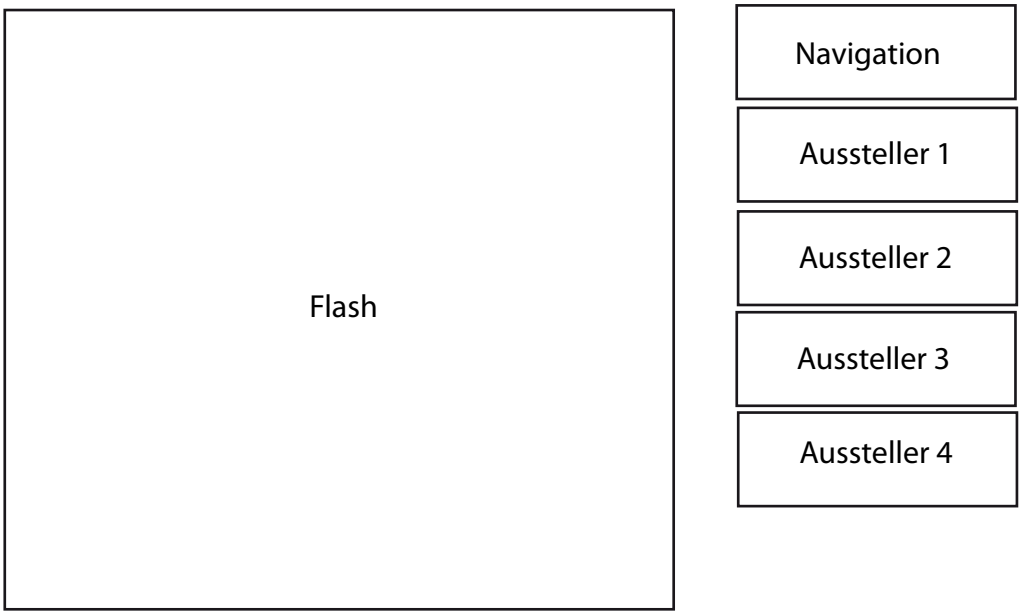


Abbildung 11: -Container

Danach werden alle benötigten Informationen aus der empfangenen XML ausgelesen.

var aussteller = root_node.childNodes[nr];
var kategorie = aussteller.childNodes[0];
var hauptkategorie = kategorie.childNodes[0];
var hauptkategorie_name = hauptkategorie.getAttribute(‚name‘);
…

Die Informationen werden in den entsprechenden Variablen gespeichert und dann auch wieder über *„innerHTML“* in die dafür erstellten Container geschrieben.

document.getElementById(‚hkat‘+nr).innerHTML = hauptkategorie_name;

Die Detailansicht eines Standes unterscheidet sich insofern, dass nur ein Ausstellercontainer erstellt werden muss, es gibt also keine Schleife. Dafür werden aber mehr Informationen aus der XML ausgelesen und dargestellt.

3.3.3.3 Kommunikation

Die Kommunikation zwischen JavaScript und ActionScript ist dank der ExternalInterface-Klasse sehr einfach und das Prinzip auch schon im Theorieteil erklärt worden (siehe 2.4.3 Kommunikation).

In unserem Projekt werden an mehreren Stellen Daten von JavaScript an ActionScript gesendet. Immer dann, wenn auf der AJAX-Seite eine Halle oder ein bestimmter Stand angeklickt wird, gescrolled wird oder über die Seitenblätterung sich die Auswahl verändert hat.

Dafür verwenden wir immer dieselbe Funktion *„rcvJS()“*, die immer 2 Parameter übermittelt. Einen für die Halle, und einen für den ausgewählten Stand.
„rcvJS(3, 9)“ entspricht somit Stand 3 in Halle 9 und *„rcvJS(9, 0)“* entspricht der Übersicht von Halle 9.

Die Kommunikation wird bei einem Klick, z.B. auf einen Aussteller, gestartet.

Nintendo of Europe GmbH

„flashmovie“ ist der Name der eingebetteten SWF.

Die übermittelten Parameter werden dann von ActionScript weiterverarbeitet (siehe 3.3.2.3 Kommunikation).

3.3.3.4 Autocomplete- Suche

Bei so vielen Hallen und Ständen, kann man schnell den Überblick verlieren, in welcher Halle welcher Aussteller ist. Der Benutzer braucht eine einfache Unterstützung um dennoch den gewünschten Aussteller zu finden. Dies haben wir mit einer Autocomplete-Suche realisiert.
Das Prinzip ist recht einfach:
Gibt der Benutzer einen Buchstaben in das Suchfeld ein, erscheinen gleich unterhalb die Aussteller, die mit diesem Buchstaben anfangen, als Vorschlagsliste.
Er kann nun aus dieser Liste den gewünschten Aussteller auswählen, oder weitere Buchstaben eintippen, wobei sich die Anzahl der Ergebnisse natürlich einschränkt.
Entscheidet sich der Benutzer nun für einen Aussteller, so werden dessen Stand oder evtl. auch Stände auf der AJAX-Seite angezeigt. Die Darstellung erfolgt über die ***„searchview.js“***, da hier im Gegensatz zur normalen Übersicht noch die Hallennummer mit ausgegeben wird. Jetzt kann der Benutzer, einen einzelnen Stand anwählen, die Flash-Ansicht springt zu diesem Aussteller und die Detailansicht wird angezeigt.

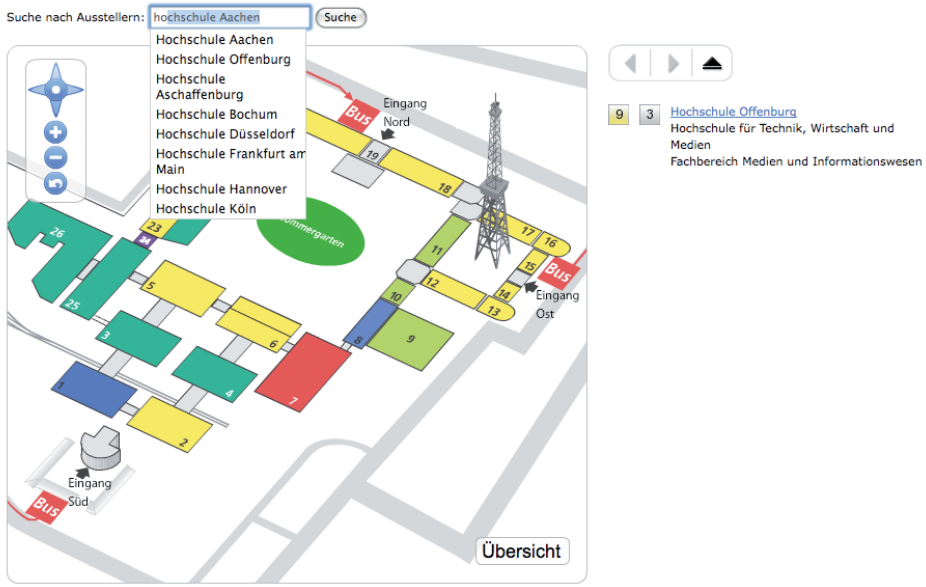


Abbildung 12: Autocomplete-Suche

Die Funktionsweise der Autocomplete- Suche ist relativ einfach.
Grundlage war ein Skript2, welches sich auf das jQuery3 Framework stützt. jQuery oder andere JavaScript Frameworks wie Prototype4 oder MooTools5 ermöglichen es, spezielle vorgefertigte Module zu verwenden, die man nicht komplett selbst schreiben muss.

Beim Laden der Seite wird einmalig eine Suchstruktur erstellt.

Zunächst senden wir einen AJAX-Request an den Server, der uns die Datei ***„gesamt.xml“*** liefert, in welcher alle Aussteller in allen Hallen aufgelistet sind.
Daraus wird ein Array (***„searchArray“***) erstellt, in dem alle Ausstellernamen eingetragen sind.

Dieses Array übergeben wir an die Funktion *„autocompleteArray()“*.

```
$(„#CityLocal“).autocompleteArray(
        searchArray,
        {
            delay:10,
            minChars:1,
            matchSubset:1,
            onItemSelect:selectItem,
            onFindValue:findValue,
            autoFill:true,
            maxItemsToShow:10
        }
    );
```

Die Notation mit *„$“* und *„#“* ergibt sich aus der Verwendung von jQuery.

Zusätzlich werden noch weitere Parameter verwendet:

„delay“: Verzögerung in Millisekunden bis zum Auslösen des Suchalgorithmus nach einer Benutzereingabe.
„minChars“: Mindestanzahl an einzugebenden Zeichen
„matchSubset“:0 oder 1 zur Unterscheidung von Untermengen.
„onItemSelect“: Funktion, die bei einer Auswahl aus der Liste aufgerufen wird
„onFindValue“: Funktion, die bei einer erfolgreichen Suche aufgerufen wird
„autoFill“: sehr wichtig, entscheidet, ob die Benutzereingaben automatisch vervollständigt werden
„maxItemesToShow“: Anzahl der sichtbaren Einträge in der Vorschlagsliste

Das Array wird nun für die Suche aufbereitet. Als Ergebnis erhält man eine Struktur, die wie folgt geschachtelt ist:

```
<A>
   <Aa>
        <Aaa>
        </Aaa>
   </Aa>
   <Ab>
   </Ab>
</A>
<B>
</B>
…
```

Innerhalb eines Tags existiert eine Liste aller Aussteller, die mit diesem Buchstaben oder der

entsprechenden Buchstabenkombination beginnen. *„<Aa>“* ist damit immer eine Untermenge von *„<A>“*.
Die Tiefe dieser geschachtelten Struktur, hängt davon ab, ab wann es für eine Buchstabenkombination nur noch ein Ergebnis gibt.
Gibt es also nur einen Eintrag, der mit „A“ beginnt, gibt es auch keine Untermengen *„<Aa>“*, *„<Ab>“*, usw.

Mit dieser Suchstruktur kann erstaunlich schnell nach möglichen Treffern gesucht werden.

Hat der Benutzer sich nun für einen Eintrag entschieden, so wird ein zweites Array innerhalb JavaScript erstellt und weitergegeben, welches dem ähnelt, das man bei einer Parameterübergabe von Flash erhält.

„param_array (1,2,3,4,5,…)“

Die Zahlen entsprechen der Stelle des gesuchten Eintrages in der *„gesamt.xml“*
Daraus kann JavaScript nun wieder eine Übersichtsansicht erstellen.

Übersicht über alle JS- Dateien.

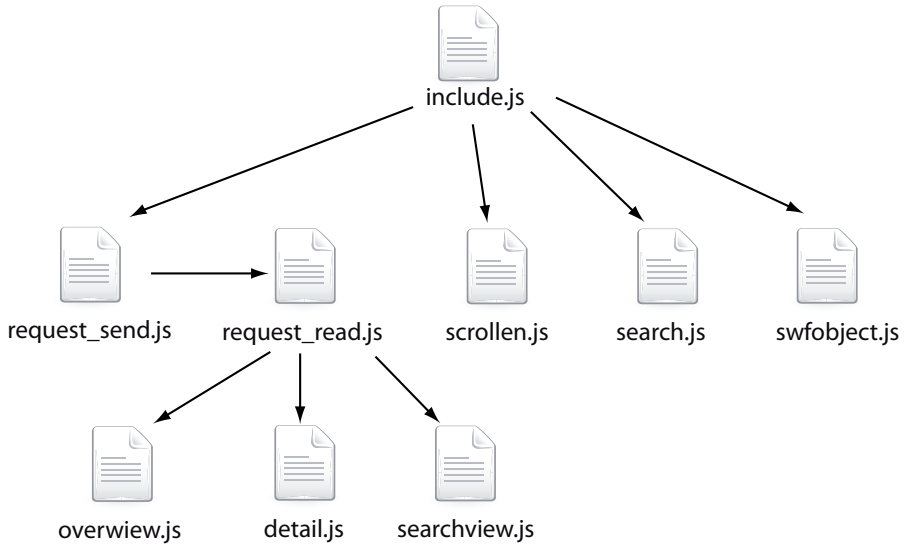


Abbildung 13: Beziehungen der Dateien in AJAX

3.3.3.5 Administration

Im Rahmen des konzeptionellen Entwurfs des Messeplans war es auch vorgesehen einen Administrativen Bereich zu schaffen, der es ermöglicht den Datenbestand der Hallen und Aussteller zu pflegen. Als grundlegende Funktionen sollten dabei die Aktionen „Neue Halle anlegen“, „Hallen ändern / löschen“, „Neuen Aussteller anlegen“ und „Aussteller ändern / löschen“ realisiert werden. Im folgenden Text wird zur Darstellung der Implementierungsweise sowie der verwendeten Techniken

die beispielhafte Aktion „Halle löschen“ beschrieben.
Wird im Administrationsbereich auf den Link „Aussteller ändern / löschen“ geklickt, wird die JavaScript-Funktion ***„sendReq(q)“*** aufgerufen, die eine asynchrone Anfrage an den Server stellt. Der Parameter ***„q“*** bestimmt, welcher Wert dem Parameter ***„action“*** im POST-Request der asynchronen Anfrage angehängt wird.
Da ein Aussteller geändert oder gelöscht werden soll, erhält der Parameter ***„action“*** den Wert ***„loadDrop“***, da zunächst ein Dropdown-Menü mit den vorhandenen Hallen geladen werden muss. Der Nutzer wählt dann aus diesem Menü die Halle aus, aus der er Aussteller löschen will. Der folgende Codeausschnitt zeigt den Teil der Funktion ***„sendReq(q)“***, der den Parameter ***„action“*** an das PHP-Skript ***„halle.php“*** sendet:

```
var req = createRequestObject();
req.open(„POST“, url, true);
req.setRequestHeader(„Content-Type“, „application/x-www-form-urlencoded; charset=UTF-8“);
(...)
req.onreadystatechange = function() {
    if(req.readyState == 4) {
        document.getElementById(„status“).innerHTML = „“;
        document.getElementById(„response“).innerHTML = req.responseText;
    }
    else
        document.getElementById(„status“).innerHTML = „lade...“;
};
req.send(„action=loadDrop“);
```

Das generieren des Dropdown-Menüs in HTML-Code wird von dem von der Funktion ***„sendReq(q)“*** aufgerufenen PHP-Skript ***„halle.php“*** übernommen. Das Skript überprüft zunächst anhand des übergebenen Wertes in ***„$_POST['action']“*** welchen Dateinamen es in die Variable ***„$file“*** schreiben soll. Diese Variable wird verwendet um der Funktion ***„loadFile($pfad)“*** bei Aufruf den richtigen Dateinamen der zu ladenden XML-Datei zu übergeben. Da der Nutzer zuerst die Halle auswählen muss, aus der er Aussteller löschen will, wird der Variablen ***„$file“*** der Wert ***„overview.xml“*** zugewiesen.
Im Anschluss wird die Funktion ***„loadFile($pfad)“*** aufgerufen, die bei Erfolg die XML-Datei als ***„DOM-Document-Objekt“*** an die Variable ***„$xml“*** zurückgibt. Sollten noch keine Hallen existieren, so existiert auch die Datei ***„overview.xml“*** nicht, und die Variable ***„$xml“*** erhält den Wert ***„null“***. Sollte dies der Fall sein, generiert das Skript einen Hinweis an den Nutzer, dass er keine Aussteller ändern / löschen kann, da bisher keine Hallen angelegt wurden. Dieser Hinweis wird dann über die JavaScript Callback-Funktion mit Hilfe des DOM in das HTML Dokument geschrieben.
Ist die XML-Datei jedoch erfolgreich geladen worden, wird die Funktion ***„createForm($xml, $drop, $action)“*** aufgerufen. Der Parameter ***„$xml“*** beinhaltet die XML-Datei als ***„DOMDocument-Objekt“***, ***„$drop“*** beinhaltet entweder den Wert ***„true“*** oder ***„false“***. Wird der Wert ***„true“*** übergeben, bedeutet dies, es muss ein Dropdown-Menü geladen werden. Der Wert im Parameter ***„$action“*** bestimmt, ob ein <***form***>-Tag generiert werden muss oder nicht. Da der Funktion ***„createForm($xml, $drop, $action)“***

im Parameter *„$drop“* der Wert *„true“* übergeben wurde, wird die Funktion *„createDropdown($xml, $action)“* aufgerufen, die aufgrund des Werts in *„$action“* das Dropdown-Menü der Hallen an die Funktion *„createForm($xml, $drop, $action)“* zurückliefert. Diese wiederum gibt aufgrund des Werts in *„$action“* das Dropdown-Menü zusammen mit einem *<form>*-Tag an die Callback-Funktion der JavaScript-Funktion *„sendReq(q)“* zurück, die es in das HTML Dokument integriert.

Wählt der Nutzer aus diesem Dropdown-Menü nun eine Halle aus, so wird erneut die JavaScript-Funktion *„sendReq(q)“* aufgerufen, die dem PHP-Skript *„halle.php“* den Wert *„loadDrop2“* übergibt, da nun das Dropdown-Menü mit den vorhandenen Ausstellern generiert werden muss aus dem der zu löschende Aussteller dann ausgewählt werden kann. Auch dieses Dropdown-Menü wird von der Funktion *„createForm($xml, $drop, $action)“* generiert, falls bereits Aussteller vorhanden sind. Falls nicht, erhält der Nutzer auch hier einen Hinweis.
Hat der Nutzer schließlich die gewünschte Ausstellernummer ausgewählt, so wird wieder die Funktion *„sendReq(q)“* aufgerufen, die dem Parameter *„action“* den Wert *„loadForm“* zuweist und diesen an das PHP-Skript *„halle.php“* übergibt. Die Funktion *„getFormValues($xml, $standNr)“* des PHP-Skripts setzt die Werte des gewünschten Ausstellers in HTML-Formularfelder und gibt diese an den Client zurück.
Wird nun der Button „Löschen“ betätigt, sendet die Funktion *„sendReq(q)“* den Wert *„delete“* an das PHP-Skript *„halle.php“*. Aufgrund dieses Wertes wird die Funktion *„delete($xml, $pfad, $standNr, $hallenNr)“* aufgerufen, die den angegebenen Aussteller aus der angegebenen Halle löscht. Sollte der letzte Aussteller gelöscht werden, wird auch die zugehörige XML-Datei gelöscht:

```
function delete($xml, $pfad, $standNr, $hallenNr) {
        global $path;
        global $file;

        $staende = $xml->getElementsByTagName(„aussteller“);

        foreach($staende as $stand) {
                $nr = $stand->getAttribute(„nr“);

                if((string)$nr == (string)$standNr) {
                        $stand->parentNode->removeChild($stand);
                        $xml->save($pfad);

                        $system = $xml->getElementsByTagName(„system“);

                        if($system->item(0)->hasChildNodes() == false){
                                $old = getcwd();
                                chdir($path);
                                unlink($file);
                                chdir($old);
```

```
                    return „<p>Aussteller Nr „.$standNr.“ in Halle „.$hallenNr.“ wurde
                    gel&ouml;scht</p>“;
                }
                else
                    return „<p>Aussteller Nr „.$standNr.“ in Halle „.$hallenNr.“ wurde
gel&ouml;scht</p>“;
            }
        }
    }
```

4. Fazit und Ausblick

Im Rückblick auf den Projektverlauf ist hinsichtlich Recherche, Konzeption und Implementierung eine wichtige Erkenntnis eindeutig festzuhalten: Beide Technologien, Flash und AJAX, können sich nicht gegenseitig ersetzen. Die Stärken der jeweiligen Technologien liegen dabei in unterschiedlichen Bereichen:

Text- und informationslastige Inhalte, bei denen die Accessibility und Usability im Vordergrund stehen, eignen sich hervorragend für den Einsatz von AJAX. Es verleiht der statischen HTML-Seite ein interaktives Verhalten und ermöglicht eine unterbrechungsfreie Kommunikation mit dem Server. An seine Grenzen stößt AJAX vor allem bei der unterschiedlichen browserinterpretation des JavaScript Codes, sowie bei multimedialen Inhalten oder individuellen Animationen. Eine Standardisierung der Browser, sowie eine Verbesserung von JavaScript sind zwei entscheidende Faktoren für die Zukunft von AJAX. Im Hinblick auf die Vorhaben der „Open-AJAX-Initative“, die ein professionelles Open-Source AJAX Framework entwickeln möchte, welches browser- und betriebssystemunabhängig sein, und sowohl die client- als auch serverseitige Programmierung einer AJAX-Applikation unterstützen soll, wird die Entwicklung von AJAX-Applikationen erheblich vereinfacht werden. Geplant ist auch die Integration dieses Frameworks in die Entwicklungsumgebung Eclipse.

Flash eignet sich in erster Linie für animierte Inhalte mit vielen Interaktionsmöglichkeiten und Multimedia-Bestandteilen. Im Vergleich zu AJAX muss bei Flash (bis auf die Einbindung des Flash-Films in das HTML-Dokument) keine Browserkompatibilität beachtet werden. In den Bereichen „Usability“ und „Accessibility“ dagegen, kommt Flash nicht an AJAX heran. Beispielsweise gibt es keine standardisierten UI-Elemente, an die der Benutzer gewöhnt ist.

Auch die prototypische Entwicklung des Messeplans im Rahmen dieser Projektarbeit zeigt, dass Flash und AJAX sich sehr gut ergänzen können. Der Flash-Teil der Arbeit ermöglicht es dem Nutzer eine detaillierte Visualisierung des Messegeländes und der einzelnen Hallen zu betrachten, wäh-

rend die textuellen Informationen mit Hilfe von AJAX wie gewohnt mit HTML dargestellt werden. Bei der Fülle von Erweiterungsmöglichkeiten dieser Implementierung wird sowohl AJAX als auch Flash hinsichtlich ihrer eigenen weiteren Entwicklung eine große Rolle spielen[32].

5. Literatur und Quellenverzeichnis

[ADOBE01] o.V.
Suchmaschienen und Flash
http://www.adobe.com/de/aboutadobe/pressroom/pr/jul2008/044.pdf
abgerufen: 26.10.2008

[ADOBE02] o.V.
Clickjacking
www.adobe.com/de/support/security/advisories/apsa08-08.html
abgerufen: 26.10.2008

[ALISTAPART] o.V.
Einbetten in HTML
http://www.alistapart.com/articles/flashsatay
abgerufen: 26.10.2008

[DOT1NE] o.V.
AJAX vs Flash
http://dot1ne.com/journal/ajax-vs-flash-round-2-arena-web20-fight
abgerufen: 26.10.2008

[GALILEO] o.V.
Accesibility
http://www.galileocomputing.de/glossar/gp/anzeige-10521/FirstLetter-A
abgerufen: 1.11.2008

[KAIJÄGER01] Jäger, Kai
AJAX in der Praxis
Springer Berlin; Seite 299-305

[KAIJÄGER02] Jäger, Kai
AJAX in der Praxis
Springer Berlin; Seite 265-274

32 [LUDWIG02]

[LUDWIG01] Diplomarbeit
„Integration von AJAX und Flash in dynamischen Webanwendungen am Beispiel des Mercedes Benz Autokonfigurators“
Sebastian Ludwig
Hochschule Offenburg, 2007
Seite 20ff

[LUDWIG02] Diplomarbeit
„Integration von AJAX und Flash in dynamischen Webanwendungen am Beispiel des Mercedes Benz Autokonfigurators“
Sebastian Ludwig
Hochschule Offenburg, 2007
Seite 98

[SELFHTML] o.V.
Einbetten in HTML
http://de.selfhtml.org/html/multimedia/objekte.htm
abgerufen: 26.10.2008

[WIKIPEDIA01] o. V.
Hypertext Markup Language
http://de.wikipedia.org/wiki/Xhtml
abgerufen: 30.10.2008

[WIKIPEDIA02] o. V.
Hypertext Markup Language
http://de.wikipedia.org/wiki/Xml
abgerufen: 30.10.2008

[WIKIPEDIA03] o. V.
Adobe Flash
http://de.wikipedia.org/wiki/Adobe_Flash
abgerufen: 19.10.2008

[WIKIPEDIA04] o. V.
Hypertext Markup Language
http://de.wikipedia.org/wiki/Cross-Site_Scripting
abgerufen: 30.10.2008

[YOUTUBE] o.V.
Animator vs Animation
www.youtube.com/watch?v=qo1d6ttbAq8
abgerufen: 28.10.2008

6. Abbildungsverzeichnis

7. Abkürzungsverzeichnis

AJAX	Asynchronous JavaScript and XML
CSS	Cascading Style Sheets
DOM	Document Object Model
E4X	ECMAScript für XML
FLV	Flash Video
GIF	Graphics Interchange Format
HE-AAC	High Efficiency Advanced Audio Coding
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
JPG	Joint Photographic Experts Group
MIME	Multipurpose Internet Mail Extensions
MSAA	Microsoft Active Accessibility
PHP	Hypertext Preprocessor
PNG	Portable Network Graphics
RIA	Rich Internet Application
SMIL	Synchronized Multimedia Integration Language
SVG	Scalable Vector Graphics
SWF	Shockwave Flash
XHR	XMLHttp-Request
XHTML	Extensible Hypertext Markup Language
XML	Extensible Markup Language
XSS	Cross-Site-Scripting

8. Eidesstattliche Versicherung

Hiermit versicheren wir, dass wir die vorliegende Projektarbeit selbstständig erarbeitet haben und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet wurden.

Offenburg, 5. November 2008

Tobias Dinter, Tobias Erdrich, David Maus und Yael Bar-Zeev