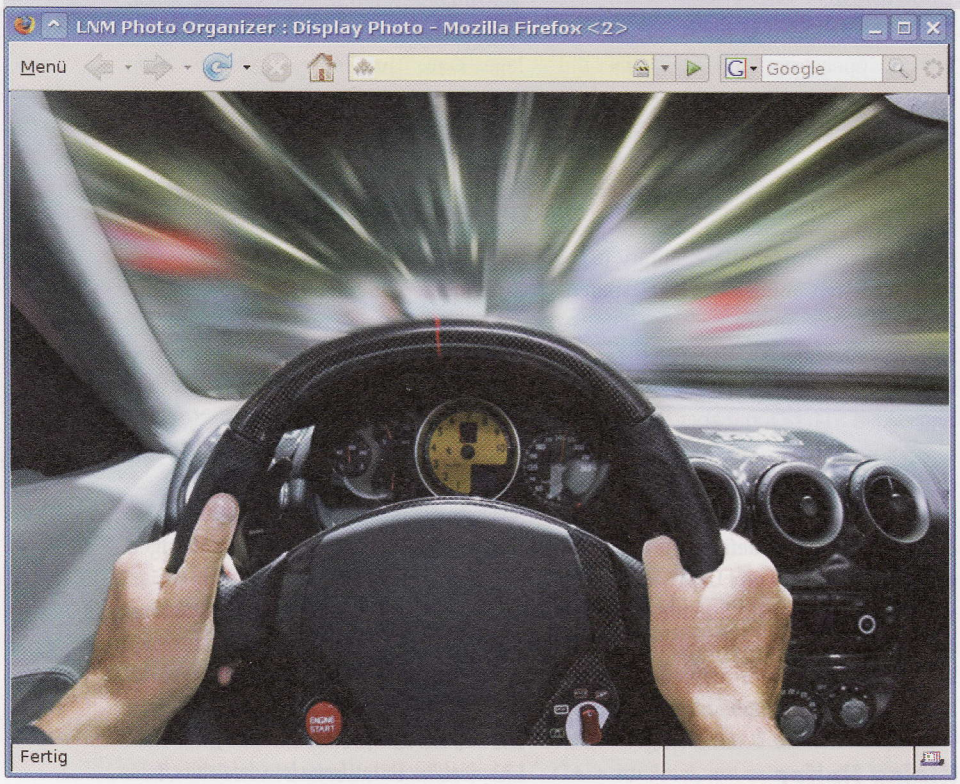
Die Technik hinter Ajax

Dynamik im Browser

Bücher und Zeitschriften waren lange Vorbild bei der Gestaltung von Webseiten. Die Navigation glich dem Umblättern: Jeder Klick auf einen Link öffnete eine neue Seite. Viele aktuelle Webpräsenzen bieten jedoch dank Ajax eine Oberfläche, die sich beinahe wie eine Desktop-Anwendung verhält. Carsten Zerbst



Seit Tim Berners-Lees ersten Webseiten haben sich die Standards für eine gelungene Präsentation im Internet stark gewandelt: Mehr und mehr orientieren sich Internerpräsenzen an Desktopanwendungen statt an Printprodukten. Hinter dem Schlagwort Ajax verbergen sich Features wie Pulldown-Menüs, sortierbare Tabellen oder interaktive Eingabeseiten. Die wesentliche Verbesserung im Vergleich zu statischen Seiten ist dabei, dass die für das Neuladen der Seiten typischen Wartezeiten entfallen.

Langer Weg

Bis eine Webseite vollständig im Browser zu sehen ist, durchlaufen Browser und Webserver mehrere Schritte, die sich zusammen zu einer für den Seitenbesucher oft unangenehmen Wartezeit summieren (Abbildung 1):

- Der Browser sendet, ausgelöst durch einen Klick auf einen Link oder die Eingabe einer URL, eine Seitenanfrage an den Webserver.
- Der Server bearbeitet die Anfrage und liefert HTML-Text und Grafiken aus. Bei hoher Last braucht er dazu oft einige Sekunden. Wie schnell die Inhalte ankommen, hängt von der Geschwindigkeit des Netzes ab. Die benötigte Zeit bleibt aber selbst in schnellen Intranets noch spürbar.
- Schließlich liest der Browser die Antwort und zeigt die Seite an. Für jedes einzelne Bild startet die gleiche Schrittfolge noch einmal.

Zusammengenommen dauern diese drei Arbeitsschritte meist etliche Sekunden, bei hoher Serverlast eventuell Minuten. Bei HTML-Seiten ohne Ajax-Technologie laufen sie bei jeder noch so kleinen Änderung auf der Seite ab.

Im Vergleich zu Rich-Client-Anwendungen leidet der Komfort dabei arg. Verzögerungsfrei aufklappende Menüs, mit einem Mausklick neu sortierbare Tabellen, Tooltipps oder Drag&Drop sind auf der Basis der zähen Seiten-Reloads nicht praxistauglich umsetzbar. HTML-Seiten, die diese Features zur Verfügung stellen, sollten daher wie lokale Programme weitgehend autonom, also ohne Rückgriff auf eine Serververbindung arbeiten.

Ohne Rückfrage

Um den Komfort zu verbessern, gehen mehr und mehr Webanwendungen dazu über, die Benutzereingaben direkt im Browser auf dem Computer des Seitenbesuchers zu verarbeiten. Von den verschiedenen Techniken, Anwenderdaten im Browser zu verarbeiten, haben sich nur zwei auf breiter Front durchgesetzt: Javascript und Flash. Beide stehen auf mehr als 90 Prozent aller Rechner zur Verfügung. Webentwickler können sie deshalb mit gutem Gewissen einsetzen. Andere Lösungen haben mit Ausnahme des auch noch verbreiteten Java-Plugins, nie ähnlich plattformübergreifende Verbreitung gefunden.

Vom Konzept her schlagen Flash und Javascript völlig unterschiedliche Wege ein. Das proprietäre Flash-Plugin führt binäre Flash-Applikationen im Browser aus. Das Plugin unterscheidet sich in seiner Einbindung in die Webseite nicht wesentlich von einer Bitmap-Grafik, bie-

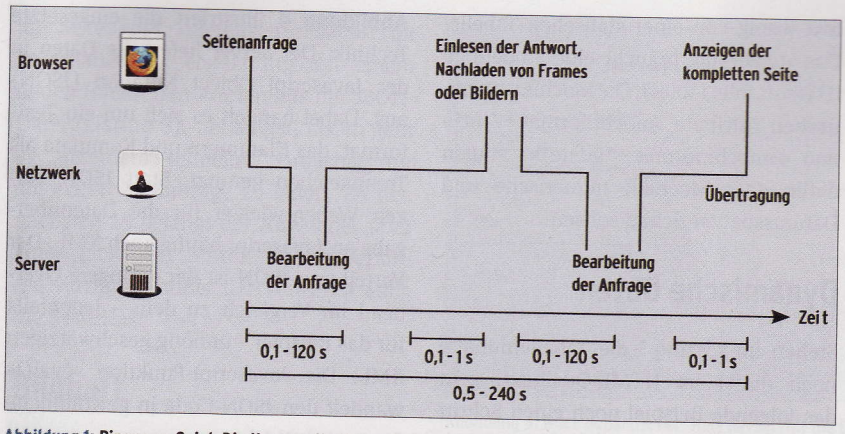


Abbildung 1: Pingpong-Spiel: Die Kommunikation zwischen Browser und Webserver, die bei Seiten ohne Ajax Voraussetzung für jede Veränderung der angezeigten Seite ist, setzt sich aus mehreren Kontakten zwischen Client (Browser) und Server zusammen und dauert in der Praxis meist etliche Sekunden.

tet dem Benutzer jedoch eine interaktive Anwendung. Die grafischen Möglichkeiten des Flash-Plugins sind ausgezeichnet. Um sie zu nutzen, führt allerdings kaum ein Weg an den Adobe-Tools vorbei. Gleichwertige Open-Source-Alternativen sind nicht in Sicht.

Dynamisch statt statisch

Javascript ist dagegen nicht auf isolierte Bereiche der Webseite beschränkt. Der im Browser integrierte Interpreter führt die Programme aus und verwandelt die ganze Seite in eine dynamisch veränderbare Oberfläche. Die Skripte erzeugen oder verändern dabei entweder den HTML-Code der Seite, passen die CSS-Stile an oder zeichnen Grafiken, die sich zur Laufzeit verändern lassen.

Die Skripte selber bestehen aus unkompiliertem Text. Anders als die Flash-Entwicklung setzt die Programmierung in Javascript daher keine besonderen Tools voraus. Ein einfacher Texteditor reicht für den Einstieg. Wie immer beim Programmieren erleichtert gutes Handwerkzeug die Arbeit. Nützlich sind zum Beispiel ein Editor mit Unterstützung für HTML, CSS sowie Javascript. Er sollte auch mit Quellcode, der diese Sprachen mischt, umgehen können ([1], [2]).

Fürs Nachvollziehen des Programmablaufs und die Fehlersuche führt kaum ein Weg an Chris Pedericks Firefox-Plugin Web Developer [3] vorbei. Es informiert über Fehler im HTML-, CSS- und Javascript-Code, gibt Aufschluss über Cookies und zeigt auch den dynamisch veränderten HTML-Code, nicht nur die vom Server ausgelieferte Version an.

Tabellen

Immer mehr Webanwendungen wie Bestellsysteme oder Warenverwaltungen bieten Webfrontends. Stücklisten oder andere listenförmige Übersichten spielen dabei eine große Rolle. Ein hilfreiches Feature ist es dabei, wenn der Benutzer die Listen sortieren kann. Basiert das Webfrontend auf statischem HTML, muss der Server die Seite neu erzeugen und übertragen. Fließenderes Arbeiten erlaubt dagegen die Client-seitige Sortierung der Daten mit Javascript.

Abbildung 2 zeigt als Beispiel ein als HTML-Tabelle realisiertes Verzeichnislisting. Per Klick auf einen Spaltenkopf soll sie sich nach der zugehörigen Spalte sortieren. Abbildung 3 erläutert, wie dies mit Hilfe von Javascript möglich ist: Eine HTML-Tabelle besteht aus verschachtelten »<tr>«- und »<td>«-Elementen, die sich über DOM [4] adressieren lassen. Javascript entfernt zunächst alle »<tr>«-Elemente – also alle Tabellenzeilen – dynamisch aus der Seite. Sie existieren dann nur noch als Inhalt eines Array in Javascript.

Die Javascript-Funktion sortiert es wie gewünscht und schreibt den Inhalt in der neuen Reihenfolge wieder zwischen die nun leeren Tags »<table>« und »</table>«. Schließlich setzt Javascript noch einen Pfeil – ein Unicode-Zeichen aus dem Bereich »Arrows« [5] – vor die Überschrift der Spalte, nach der die Tabelle nun sortiert ist.

Baukasten-System

Das geschilderte Verfahren lässt sich mit einigen Hundert Zeilen umsetzen. Einfacher ist jedoch der Rückgriff auf fertige Lösungen. Im Internet findet sich ein gutes Dutzend verbreiteter Javascript-Bibliotheken. Sie enthalten Funktionen für häufige Aufgabenstellungen wie das Sortieren von Tabellen, das Erzeugen abgerundeter Ecken in HTML und das Zeichnen von Baumgrafen. Für diese Bi-

Rechte	Typ	Eigentümer	Gruppe	Größe↓	Änderung	Name
lrwxrwxrwx	1	root	root	18	11.02.08	printcap
-rw-r--r--	1	root	root	149	26.11.06	hosts.deny
-rw-r--r--	1	root	root	188	26.11.06	hosts.equiv
-rw-r--r--	1	root	root	191	26.11.06	hosts.lpd
-rw-r--r--	1	root	root	530	27.01.07	group.YaST2save
-rw-r--r--	1	root	root	536	27.01.07	group.old
-rw-r--r--	1	root	root	551	27.01.07	group
-rw-r--r--	1	root	root	677	19.12.06	hosts.YaST2save
-rw-r--r--	1	root	root	715	27.12.06	hosts
-rw-r--r--	1	root	root	1357	02.10.08	passwd
-rw-r--r--	1	root	root	2639	26.11.06	hosts.allow

Abbildung 2: Wohlgeordnet: Ein Klick auf einen Tabellenkopf genügt, und das Client-seitige Javascript sortiert die Tabelle nach der gewünschten Spalte, ein verzögernder Kontakt zum Server ist dazu nicht nötig.

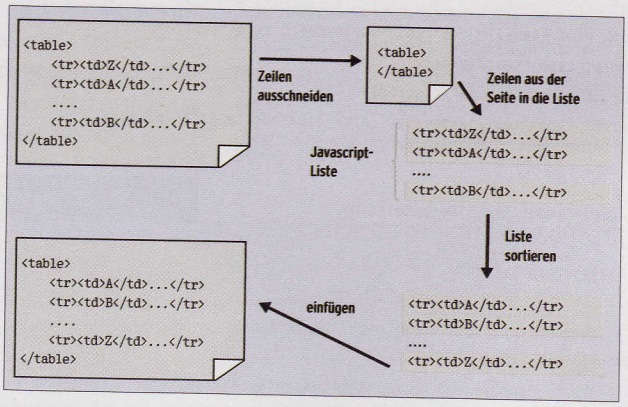


Abbildung 3: Javascript sortiert Tabellen, indem es alle Zeilen dynamisch aus der Tabelle entfernt, sie in einem Array zwischenspeichert, dort sortiert und in der neuen Reihenfolge wieder einfügt.

bliotheken spricht im Vergleich mit eigenen Lösungen außer ihrem inzwischen großen Funktionsumfang, dass sie Kompatibilität mit den gängigen Browsern garantieren. Obwohl Javascript ein ECMA-Standard ist [6], unterscheiden sich die Implementierungen in den Browsern unter Linux, Mac OS X und Windows doch noch in wesentlichen Punkten. Die meisten Bibliotheken kapseln diese Unterschiede und erleichtern so das Leben der Entwickler.

In die Praxis

Listing 1 zeigt den HTML-Code für eine sortierbare Tabelle auf Basis der freien Bibliothek Mochikit [7]. Außer den zwei Includes in Zeile 6, die Mochikit einbinden, unterscheidet sich der HTML-Quelltext wenig von einer statischen Tabelle. Das »table«-Tag braucht eine eindeutige ID (»sortable_table«). Die Mochikit-spezifischen Attribute »mochi:format = "int"« und »mochi:format = "gdate"« sorgen dafür, dass Mochikit numerische und Datumsspalten richtig sortiert.

Dynamische Daten

Stehen im Listing 1 die Tabelleninhalte noch direkt im HTML-Quelltext, geht das folgende Beispiel noch einen Schritt weiter. Die Anwendung holt die Tabelleninhalte nun ohne Neuladen der Seite vom Server. Über Links, Buttons oder Menüpunkte lässt sich die Tabelle so mit unterschiedlichen Werten füllen. Auch Suchergebnisse blendet die Anwendung ohne Seiten-Reload ein.

Abbildung 4 illustriert die eingesetzte Technik: Der Server liefert die Daten in der Javascript Object Notation (JSON) aus. Dabei handelt es sich um ein Textformat, das Klammern und Kommata als Trennzeichen benutzt. Statt JSON nutzen Webentwickler für die Datenübergabe an Javascript häufig auch XML. Der Vorteil von JSON ist der geringere Overhead im Vergleich zu dem – jedenfalls für das Beispiel – unnötig geschwätzigen XML. Die Javascript-Funktion »eval()« wandelt den JSON-Code in gewöhnliche Javascript-Objekte um.

Auf Benutzeranfrage

Listing 2 zeigt den HTML-Code. Statt des Tabelleninhalts steht hier nur ein »tbody«-Element als Platzhalter. Das

Listing 1: Sortierbare Tabelle

```
01 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
02 <html>
03     <head>
04         <title>Sortierbare Tabelle</title>
05         <meta http-equiv="Content-Type" content="text/html;
   charset=UTF-8">
06         <link href="tabelle.css" rel="stylesheet" type="text/css" />
07         <script type="text/javascript" src="lib/MochiKit/MochiKit.js">
   </script>
08         <script type="text/javascript" src="sortierbareTabelle.js">
   </script>
09     </head>
10     <body>
11         <TABLE  id="sortierbareTabelle"  class="datagrid">
12             <THEAD>
13                 <TH>Rechte</TH> .... <TH mochi:format="int">Größe</TH>
   <TH mochi:format="gdate">Änderung</TH>
14             </THEAD>
15             <TBODY>
16                 <TR>
17                     <TD>-rw-r--r--</TD><TD>1</TD><TD>root</TD>
   <TD>root</TD><TD>551</TD><TD>27.01.07</TD><TD>group</TD>
18                 </TR>
19                 <TR>
20                     ...
21                 </TR>
22             </TBODY>
23         </TABLE>
24     </body>
25 </html>
```

Listing 2: Dynamisch erzeugte Tabelle

```
01 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
02 <html>
03     <head>
04         <link href="tabelle.css" rel="stylesheet" type="text/css" />
05         <script type="text/javascript"
   src="lib/MochiKit/MochiKit.js"></script>
06         <script type="text/javascript" src="AjaxTabelle.js"></script>
07     </head>
08     <body>
09         <a href="index.html">Beispiele</a>
10         <hr>
11         <h4>Ajax Tabelle</h4>
12         <p>
13             <a href="top.json" mochi:dataformat="json">
   Neuladen Tabelle 1</a><br>
14             <a href="top2.json" mochi:dataformat="json">
   Neuladen Tabelle 2</a>
15         </p>
16         <table id="sortable_table" class="datagrid">
17             <thead>
18                 <tr>
19                     <th mochi:sortcolumn="PID int">PID</th>
20                     <th mochi:sortcolumn="USER str">USER</th>
21                     [...]
22                     <th mochi:sortcolumn="COMMAND str">COMMAND</th>
23                 </tr>
24             </thead>
25             <!-- wird durch den Inhalt der JSON Datei ersetzt -->
26             <tbody class="mochi-template">
27                 <tr mochi:repeat="item domains">
28                     <td mochi:content="item.PID"></td>
29                     <td mochi:content="item.USER"></td>
30                     [...]
31                     <td mochi:content="item.COMMAND"></td>
32                 </tr>
33             </tbody>
34         </table>
35     </body>
36 </html>
```

im Seitenkopf referenzierte Javascript »AjaxTabelle.js« ersetzt ihn mit einem neuen Inhalt, den Werten aus der JSON-Datei (Listing 3). Bei langen Listen geht es außerdem schneller, zunächst nur die ersten 25 Einträge zu laden. Zu den nächsten 25 gelangt der Anwender über einen Link oder Button. Dieses Verfahren verringert außer der Wartezeit auch die Serverlast.

Bildhaft

Die bisherigen Beispiele beschränkten sich auf einfache HTML-Elemente wie Listen und Tabellen. Zusammen mit Cascading Stylesheets lassen sich in Javascript auch Widgets und Funktionen programmieren, die in HTML gar nicht vorgesehen sind. Mit Hilfe einer Abfrage der Mausposition lassen sich zum Beispiel Tooltipps und sogar Drag&Drop realisieren. Fertige Implementierungen sind in den meisten Javascript-Bibliotheken bereits enthalten.

Die Möglichkeiten von dynamischem HTML sind damit noch nicht ausgeschöpft. Sie reichen bis hin zu grafischen Darstellungen, die in klassischem HTML nur durch eingebundene Bitmaps lösbar waren. Abbildung 5 zeigt eine mit Javascript gezeichnete Baumstruktur.

Solche Client-seitig zur Laufzeit erzeugten Grafiken weisen im Vergleich zu Bitmaps Vorteile auf. Sie liegen nicht in einer festen Auflösung vor und können sich damit der im Browser eingestellen Fontgröße anpassen. Benutzereingaben lassen sich mit ihnen ohne Server-Interaktion visualisieren, was die Serverlast verringert. Normalerweise geht der größte Teil der Netzbandbreite bei einem Webserver auf das Konto der zu übertragenden Grafiken.

HTML-Erweiterungen

Ein leistungsfähiges Verfahren, um mit Javascript Grafiken zu erzeugen, sind die XML-basierte Grafiksprachen SVG [8], Canvas [9] und Vector Markup Language [10]. Sie lassen sich wie normale Grafiken in HTML einbetten, bieten aber die Möglichkeit, typische Grafikelemente wie Linien, Flächen oder Text zur Laufzeit hinzuzufügen. Damit können sie ebenso interaktiv auf Benutzereingaben

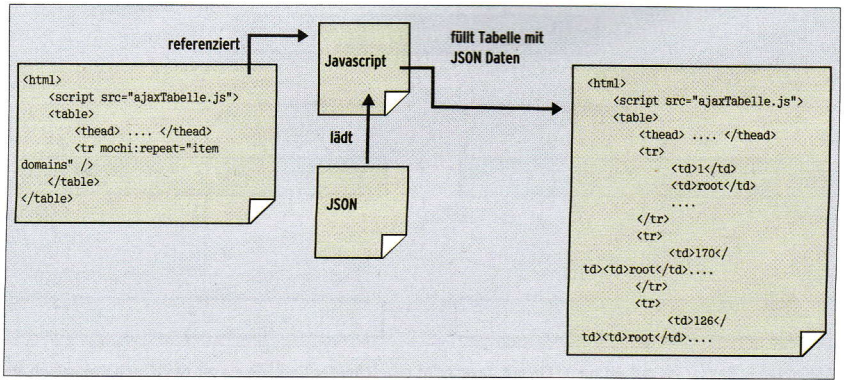


Abbildung 4: Noch dynamischer: Holt Javascript die Daten im JSON-Format vom Server, lassen sich Tabellen nicht nur neu sortieren, sondern auch ohne Neuladen der Seite mit anderen Daten füllen.

reagieren wie dynamisch über Javascript verändertes HTML. Zeichenprogramme wie Inkscape [11] oder Karbon [12] helfen beim Entwurf.

Leider hapert es noch mit der Browserübergreifenden Verfügbarkeit: Keiner der verbreiteten Browser unterstützt alle drei genannten Formate. Firefox beherrscht SVG und Canvas, Safari ausschließlich Canvas, der Internet Explorer lediglich VML. Für den Internet Explorer gibt es zwar zwei Javascript-Bibliotheken von Google bei SVG [13] beziehungsweise Canvas [14], letztlich ist die Abdeckung der Windows-Plattform aber nicht zuverlässig gewährleistet.

Bewährte Bordmittel

Viele Diagramme oder Grafen lassen sich aber über Javascript auch mit den Bordmitteln HTML und CSS erzeugen. Der unter [15] bereitgestellte Code zeichnet den Baumgrafen aus Abbildung 5. Die Kästen bestehen aus für CSS frei positionierten »div«-Elementen. Wenn beim Platzieren und Festlegen der Größe die Maßeinheit »em« statt »px« (Pixel) zum Einsatz kommt, orientieren sich Höhe, Breite und Position der Kästen an der Breite des Buchstabens „m“. Der ganze Graf skaliert dann passend zur Textgröße, ohne dass sich der Entwickler darum kümmern muss. Der Anwender kann die Grafik in den meisten Browsern zudem bequem mit dem Mausrad vergrößern oder verkleinern.

Etwas schwieriger ist das Zeichnen der Verbindungslinien. Da Javascript keine grafischen Elemente wie Linien oder Kreise kennt, hilft hier nur der Trick, die Grafikelemente Pixel für Pixel als kleine »div«-Elemente zu zeichnen. Walter Zorns JSGraphics [16] kapselt diesen aufwändigen Prozess. Für Entwickler stehen Grundformen wie Linien, Kreise oder Polygonflächen bereit.

Die Funktion »verbinde()« nutzt diese, um zwei Kästchen mit Linien zu verbinden. Sie ermittelt außerdem die Positionen der Kästen zur Laufzeit, sodass bei Veränderungen der Schriftgröße die ganze Grafik skaliert. Die Funktion »hierachieAnzeigen()« malt die Verbindungslinien dann neu. Lawrence Carvalhos Textresizer [17] ruft diese Funktion immer dann auf, wenn der Benutzer die Schriftgröße verändert.

Das Ergebnis ist ein verlustlos zoombares Ajax-Widget für Baumstrukturen, wie es sich mit Bitmaps nicht realisieren lässt. Da es ausschließlich auf HTML-

Listing 3: JSON-Daten

```
01 {
02     "columns": [ "PID", "USER","PR","NI", "VIRT", "RES", "SHR", "S", "CPU","MEM", "TIME", "COMMAND"],
03     "rows": [
04         [ "6620","cz","15","0","166912","57344","40960", "S","11.6","5.6", "0:02.40", "soffice.bin"],
05         [ "3701","root","15","0","241664","225280","17408", "S","4","21.8", "4:23.50", "X"],
06         [ "4496","cz","15","0","63828","20480","15360", "R","2","2", "0:16.02", "gnome-panel"],
07         [ "4506","cz","15","0","70480","18432","10240", "R","2","1.8", "0:04.14", "gnome-terminal"],
08     ]
09 }
```

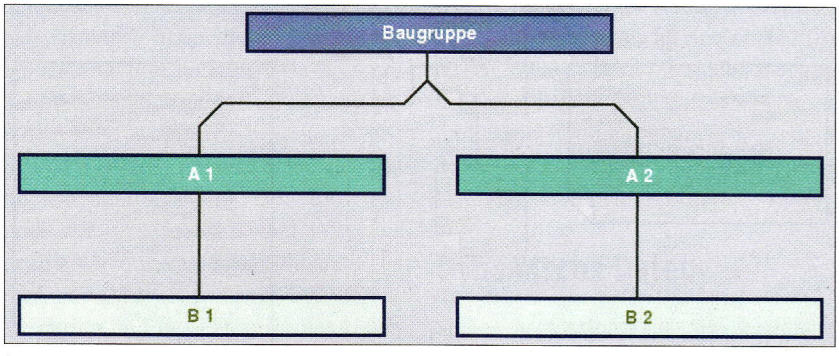


Abbildung 5: Besser als mit Bitmaps: Ein mit Javascript gezeichneter Graf wie dieser passt sich dynamisch an Änderungen der Schriftgröße an. Außerdem spart er die für Bilder benötigte hohe Bandbreite ein.

Elementen beruht, sind keine nicht Browser-übergreifenden Erweiterungen wie Canvas oder VML nötig. In Browsern ohne Javascript- und CSS-Unterstützung ist wenigstens der Kastentext lesbar.

Für und Wieder

Bei der Benutzerfreundlichkeit gewinnen viele Seiten durch den Einsatz von Javascript und Ajax: Kurze Antwortzeiten und eingesparte Seitenreloads sorgen für Sympathiepunkte bei den Besuchern. Allerdings treten durch den Einsatz von Ajax auch Probleme auf, die bei statischen Seiten nicht bestehen: So kann der Anwender normalerweise nicht mit den Vor- und Zurück-Buttons navigieren. In diesem Punkt durchkreuzen mit Javascript dynamisch veränderte Seiten die Benutzererwartungen.

Das Problem tritt bereits bei einfachen dynamischen Menüs auf: Hat der Anwender per Mausklick zum Beispiel ein Untermenü aufgeklappt, bringt ihn der Zurück-Button nicht zum früheren Seitenstatus zurück, sondern öffnet die vorher besuchte Seite. Bei einem Menü stört dies noch wenig, doch es verwirrt, wenn das Client-seitige Javascript die Seite so verändert, dass sie auf den Benutzer wie eine neue Seite wirkt.

Übersehen

Der Browser kann die wechselnden Zustände einer Ajax-Seite nicht erkennen, da die URL konstant bleibt. Deshalb fangen auch Lesezeichen nicht den Status der Seite ein. Bei einer Website, deren Navigation auf Ajax basiert, landet der Anwender daher über Bookmarks zwangsläufig auf der Startseite.

Es ist also zunächst abzuwägen, ob eine schnell reagierende Seite oder eine wie gewohnt funktionierende Browse-History und die Möglichkeit, Unterseiten in Bookmarks abzulegen, dem Anwender mehr Vorteile bringt. Während Clientseitiges Javascript, das ohne Seiten-Reload Menüs ausklappt oder Tabellen umsortiert, eine elegante Lösung ist, würde sich die Benutzbarkeit einer Shop- oder Katalogseite mit Hunderten Unterseiten drastisch verschlechtern, wenn der Benutzer auf Bookmarks und die Vor- und Zurück-Buttons verzichten müsste.

Reparaturmaßnahmen

Allerdings gibt es für das Browse-History- und Bookmark-Problem Workarounds: Google-Maps etwa stellt oberhalb der Karten-Darstellungen eine alternative URL zur Verfügung, die den Kartenausschnitt in GET-Parametern festhält. Andere Workarounds nutzen den HTML-Anker, der normalerweise dazu dient, in einer URL bestimmte Punkte auf einer Seite festzuhalten.

Der Anker ist jener Teil der URL, der nach einem »#«-Zeichen folgt. Wie die gesamte Seitenadresse lässt er sich über Javascript verändern, die Seite wird dabei nicht erneut geladen. Findet der Browser für die Zeichenfolge hinter dem Doppelkreuz in der Seite kein entsprechendes Anker-Tag, bleibt die Anzeige unverändert.

Der Anker lässt sich also ausgezeichnet zum Zwischenspeichern von Statusinformation einsetzen: Der Anker-Bereich ist der einzige Teil der URL, der sich verändern lässt, während die Seite unverändert geladen bleibt. Ein Linux-Magazin-Artikel erläutert die Details [18].

Die Möglichkeiten von mit Javascript erzeugten Grafiken fallen allerdings im Vergleich mit Flash sehr bescheiden aus. Mit HMTL und CSS lassen sich nur Rechtecke und Text darstellen. Bibliotheken wie die JSGraphics von Walter Zorn fügen weitere Formen wie Kreise und Polygone hinzu. Weiter gehende grafische Fähigkeiten bieten in die Webseite eingebettete SVG-Grafiken oder Canvas. Letzteres ist ein Element des künftigen Webstandards HTML 5. Beide eignen sich jedoch kaum für öffentlich zugängliche Internetpräsenzen, da sie gegenwärtig nicht für alle gängigen Browser zu Verfügung stehen. *(pkr)*

Infos

[1] Jedit: [http://www.jedit.org]
[2] Netbeans: [http://www.netbeans.org]
[3] Webdeveloper-Firefox-Plugin: [http://chrispederick.com/work/web-developer]
[4] Domain Acces Model: [http://www.w3.org/DOM]
[5] Unicode-Pfeile: [http://www.alanwood.net/unicode/arrows.html]
[6] Javascript Standard: [http://www.ecma-international.org/publications/standards/Ecma-262.htm]
[7] Mochikit: [http://www.mochikit.com]
[8] SVG: [http://www.w3.org/Graphics/SVG]
[9] HTML-Canvas: [http://www.w3.org/html/wg/html5/#the-canvas]
[10] VML: [http://www.w3.org/TR/1998/NOTE-VML-19980513]
[11] Inkscape: [http://www.inkscape.org]
[12] Karbon: [http://www.koffice.org/karbon]
[13] SVG2VML: [http://code.google.com/p/svg2vml]
[14] Explorer Canvas: [http://excanvas.sourceforge.net]
[15] Javascript-Baumgraf: [ftp://ftp.linux-magazin.de/pub/magazin/2008/05/Ajax/baumgraf.html]
[16] JSGraphics: [http://www.walterzorn.de/jsgraphics/jsgraphics.htm]
[17] Text Resizer: [http://www.alistapart.com/articles/fontresizing]
[18] Back-Button und Bookmarks mit Ajax: Tobias Hauser, „Ajax-Probleme", Linux-Magazin 02/2007, S. 46

Der Autor

Carsten Zerbst entwickelt Individual-Software im CAD- und PDM-Umfeld für die Luft- und Raumfahrtindustrie und den Schiffbau.