Einfach für Alle
Das Angebot der Aktion Mensch für ein barrierefreies Internet

# Konzeption: Toleranz und Rücksicht einplanen

Teil 1 unserer Serie zu barrierefreien Formularen

Stand: 14.06.2011, Autor: tc

## Woran erkennt man ein gutes Formular?

Formulare sind so etwas wie Zahnärzte oder Tankstellen: keiner will hin, aber alle müssen irgendwann mal. Aber genauso wie Tankstellen versuchen, die lästige Pflichtübung in ein angenehmes Erlebnis zu verwandeln, so sollten Ihre Formulare alles andere als langweilig sein oder dem Nutzer unnötige Aufgaben aufbürden. Checklisten, wie solche Fehler zu vermeiden sind, gibt es zuhauf im Netz, nur: woran erkennt man ein wirklich gutes Formular?

Jessica Enders formuliert die folgenden vier Kriterien: »Clear, Concise, Clever, Co-operative« – **Klar, Knapp, Klug, Kooperativ** – diese positiven Eigenschaften sind es, die Nutzer von einem Formular erwarten dürfen; dazu kommt noch die gerade für komplexe Abläufe wichtige Eigenschaft der Konsistenz.

**– Klar** bedeutet, dass der Nutzer mit minimalem Aufwand herausfinden kann, was von ihm verlangt wird. Einfluss auf die Klarheit eines Formulars haben z.B. die verwendete Ansprache der Nutzer, das Formular-Layout und die Optionen, die dem Nutzer zur Verfügung stehen. Die Herausforderung für den Webdesigner: die Ziele des Formulars mit den Ansprüchen der Nutzer in Einklang zu bringen. Dafür müssen Sie die Bedürfnisse der Nutzer und den Nutzungskontext kennen.

Die 5 K's guter Formulare:

- **Klar**
- **Knapp**
- **Klug**
- **Kooperativ**
- **Konsistent**

**– Knapp** bedeutet die benötigten Informationen auf eine möglichst effiziente Art zu sammeln. Viele Abfragen (und damit die Anzahl der benötigten Formularfelder) können durch eine genaue Formulierung reduziert werden. Dabei heisst »knapp« nicht zwangsläufig »kurz«. Manchmal kann es sinnvoll sein, zusätzliche Zwischenschritte einzufügen, um auf bestimmte Eingaben des Nutzers zu reagieren und ihm z.B. die Möglichkeit zu geben, Daten zu präzisieren.

Ein Beispiel für ein Formular, das man deutlich knapper gestalten könnte, sehen Sie im folgenden Screenshot: Auch in einer Millionenstadt ist bei den wenigsten Adressen eine solch präzise Abfrage von Stiege, Stock und Tür notwendig, als dass dies grundsätzlich immer und bei allen Nutzern abgefragt werden sollte. Hier hätte man sich besser für ein gemeinsames Feld »Weitere Adresszusätze« oder eine Funktion zur Einblendung der zusätzlichen Optionen entschieden und damit das Formular für die Mehrzahl der Anwendungsfälle deutlich verschlankt.


**– Klug** bedeutet, dass die kognitive Belastung für den Nutzer reduziert und er nicht vor unlösbare Aufgaben gestellt wird. Die einfachste Art, diese Belastung zu reduzieren ist, die Aufgaben sequentiell und in einer sinnvollen Reihenfolge zu stellen. Online-Formulare haben gegenüber ihren Print-Pendants den enormen Vorteil, dass sie nicht alle möglichen Eventualitäten auf dem gleichen Stück Papier abfangen müssen. Daher sollten intelligente Formulare nur die unbedingt erforderlichen Daten abfragen (viel mehr dürfen Sie aus Gründen des Datenschutzes sowieso nicht) und optionale oder an Bedingungen geknüpfte Eingaben erst dann zeigen, wenn sie benötigt werden. So macht zum Beispiel die Abfrage der Anzahl der schulpflichtigen Kinder keinen Sinn, wenn der Nutzer vorher bei der Anzahl der Kinder eine Null eingetragen hat.

**– Kooperativ** verhalten sich Formulare, die mit dem Nutzer arbeiten, nicht gegen ihn. Kooperative Formulare erfül-

len die in sie gesetzten Erwartungen, sie entsprechen dem mentalen Modell, das der Nutzer sich von der Anwendung gebildet hat, sie erklären sich selbst und die einzugebenden Daten, sie geben Hintergrundinformationen und Hilfen. Neben diesen auch auf gedruckte Formulare zutreffenden Kriterien gilt für Online-Formulare, dass sie dem Nutzer vor der endgültigen Datenübergabe die Möglichkeit zur Korrektur geben und dass sie frei von Software-Bugs sind und stabil laufen.

Weder kooperativ noch sonderlich klug erscheint uns das Formular im folgenden Screenshot eines DSL-Verfügbarkeits-Checks: Die Angabe der Hausnummer »13 a« wird vom System mit der Fehlermeldung »Bitte nur Zahlen (0 bis 9) eingeben« quittiert. Dies lässt nur einen Schluss zu: das anbietende Unternehmen ist offensichtlich nicht an Neukunden interessiert, die in Hausnummern größer als 9 wohnen und zu allem Überfluss auch noch einen Buchstaben in der Hausnummer haben.


**– Konsistent** sind Formulare, die einmal gesetzte Regeln konsequent durchhalten und für den Nutzer vorhersehbar sind. Konsistenz bezieht sich dabei auf alle Ebenen des Formulars, d.h. von der Ansprache des Nutzers über die auszuführenden Aktionen bis zum Layout und dem Erscheinungsbild der Kontroll- und Eingabeelemente. Formulare, die die Ansprache des Nutzers wechseln, sind nicht konsistent. Beim folgenden Screenshot eines Anmeldeformulars wird der Nutzer teils in der ersten Person angesprochen (»Andere können mich über meine E-Mail-Adresse finden«, »Mein Konto erstellen«, »Ich möchte Insider-Infos« etc.), dann wechselt der Text an mehreren Stellen in die zweite Person (»Dein öffentliches Profil«, »Wenn Du "Mein Konto erstellen" anklickst« etc.). und das Formular ist nicht vollständig lokalisiert (*»Your full name will appear on your public profile«, »Printable version«* etc.)


Menschliche Nutzer agieren jetzt und in Zukunft auf der Basis von Erfahrungen aus der Vergangenheit, seien diese positiv oder negativ. Ein konsistentes Formular sollte sich der erlernten Arbeitsweise des Nutzers bestmöglich anpassen.

Stattdessen treffen Nutzer laufend auf:

- Formulare, die zu lang und zu kompliziert sind, die ein Übermaß an kognitiver Aktivität verlangen und den Fluss unterbrechen;
- Formulare, die nicht klar sind und deren exakter Zweck nicht bestimmt werden kann;
- Formulare, die den Nutzer zwingen, bestimmte Fragen zu beantworten, auch wenn diese für den Nutzer irrelevant sind;
- Formulare, die dem Nutzer die Kontrolle über das Geschehen entziehen und
- Formulare, die den Nutzer »anschreien« wenn man sie nicht korrekt ausfüllt.

Diese Formulare sind für viele Nutzer lästig aber mehr oder weniger nutzbar. Menschen mit Behinderung treffen auf Probleme, die eine Nutzung unmöglich machen. Eine Bestellung kann nicht durchgeführt werden, wenn der blinde Nutzer den »Bestellen«-Knopf nicht finden kann. Die Beteiligung an einer Diskussion ist für einen sehbehinderten Nutzer nicht möglich, wenn er mit seiner hochgradig auf seine Bedürfnisse angepassten Konfiguration das Kommentar-Formular nicht wahrnehmen kann. Ein Nutzer mit einer motorischen Behinderung wird aufgeben, wenn er mikroskopisch kleine Formularelemente und Icons nicht zielsicher treffen kann und ein Nutzer mit kognitiver Behinderung wird sich nicht durch eine ellenlange, mehrere Seiten dauernde Folge von Formularfeldern kämpfen, deren Sinn sich ihm nicht erschließt; langatmige und im Juristendeutsch verfasste Erklärungstexte und AGBs verschrecken all jene mit einer den Ansprüchen der Site nicht genügenden Schriftsprachkompetenz.

Eine Liste mit typischen Behinderungen und ihren Auswirkungen auf die Web-Nutzung finden Sie beim AEGIS Consortium unter dem Stichwort »Personas« und im auch online verfügbaren Buch »Just Ask« von Shawn Lawton Henry bei »Personas in User-Centered Design«. Die Erkenntnisse aus diesen »Personas« genannten typischen Nutzungsszenarien sollten Sie bereits in die Konzeption Ihrer Anwendung einfließen lassen.

Die Konsequenzen aus nicht barrierefreien Formular-Anwendungen für viele Menschen mit Behinderung sind damit klar: Kein Einkauf oder keine Beteiligung am öffentlichen Leben.

## Klug konzipierte Prozesse

Gutes Design ist mehr als nur eine bunte Dekoration und beginnt bereits in der Konzeption. Ein gutes Design nimmt sich selbst zurück und unterstützt den Nutzer in der Bewältigung der Aufgabe. Niemand käme auf die Idee, Formularen einen hohen Spaßfaktor zu bescheinigen, aber gerade weil niemand Formulare ausfüllen mag, ist ein gutes Design der Formulare unumgänglich, wenn sie denn benutzt werden sollen.

Wobei – ein Formular muss ja nicht immer aussehen wie eine Einkommenssteuererklärung oder so beschränkte Funktionen haben wie ein Kontaktformular. Ohne Formulare gäbe es kein Mitmach-Web, denn Angebote wie Twitter, flickr oder Wikipedia sind ohne die Funktionen von Formularelementen gar nicht möglich.


Gerade bei Formularen bestehen eine ganze Reihe Barrieren für Menschen mit Behinderungen: unnötige Eingaben oder die mehrfache Abfrage von bereits vorliegenden Daten sind für Nutzer mit motorischer Behinderung nicht nur lästig, sondern eine echte Hürde. Wenn sich ein Kunde Ihnen gegenüber z.B. schon mit seiner Kundennummer identifiziert hat, dann ist die erneute Abfrage von persönlichen Daten für diese Nutzer sehr zeitraubend; verbunden mit der Gefahr, dass in der Zwischenzeit die Session abläuft und der Benutzer wieder ganz von vorne anfangen muss.

In der Musik-Community last.fm findet sich ein Beispiel: nach Auswahl der Zeitzone (also MESZ oder GMT) stehen in der darauf folgenden Länderauswahl noch Länder und Kontinente, die von dieser Zeitzone aus gesehen auf der anderen Seite des Planeten liegen. Nutzern, die auf die Tastaturbedienung angewiesen sind, aber in einem Land wohnen, dessen Anfangsbuchstabe im letzten Drittel des Alphabets ist, steht nun eine Odyssee durch eine ellenlange Liste bevor. Auch wenn geübte Nutzer kurzerhand den Anfangsbuchstaben drücken: steht Deutschland denn nun unter Germany, Federal Republic of Germany, Bundesrepublik Deutschland oder Deutschland?

Für sehbehinderte Nutzer ist die Orientierung in komplexen Formularen besonders wichtig. Diese Nutzer haben oft sehr individuelle Einstellungen. Die Erfahrung aus den BIENE-Tests der vergangenen Jahre hat gezeigt, dass von der Standardkonfiguration abweichende Einstellungen wie veränderte Schriftgröße, eigene Farbeinstellungen etc. in vielen Designs nicht berücksichtigt werden.

**Tipp:** wenn sich Ihr Formular hauptsächlich an ein deutschsprachiges Publikum richtet, dann sollten Sie die Optionen eventuell unterteilen in einen ersten Teil mit deutschsprachigen Ländern (D/A/CH/…) und einen zweiten Teil mit dem Rest.

Nutzern mit kognitiven Behinderungen erleichtern Sie das Ausfüllen, indem nicht nur das Formular selbst auf das Nötigste reduziert wird, sondern auch der Rest der Seite, in die das Formular eingebettet ist. Der Nutzer nimmt meist nichts anderes als das Formular auf der Seite wahr – dann können Sie unnötiges Beiwerk auch gleich ganz weglassen und damit eine kognitive Überfrachtung Ihrer Anwendung verhindern. Klar erkennbare Abläufe minimieren den Lernaufwand und helfen allen Nutzern.

Vor den Details der Formulargestaltung stehen einige grundsätzliche Fragen, mit denen Sie bereits in der Konzeptions- und frühen Designphase viele spätere Probleme Ihrer Nutzer verhindern können. Stellen Sie sich folgende Fragen vor der Umsetzung:

- Sind alle Abfragen wirklich nötig? Warum stellen Sie diese Fragen?
- Ist die Abfrage zum jetzigen Zeitpunkt überhaupt angebracht?
- Sind die abgefragten Daten schon vorhanden? Gibt es andere Wege, um an diese Daten zu kommen? Muss der Nutzer hier Ihre Arbeit machen?
- Können Sie optionale Fragen ausblendbar machen? Sind wirklich alle Abfragen zwingend notwendig oder kann man diese auch weglassen? Ein Beispiel: wenn Rechnungs- und Versandadresse identisch sind, dann sollten Sie den Nutzer nicht zwingen, beides auszufüllen, sondern stattdessen eine Option zur Übernahme der Daten

anbieten.

Wenn Sie Formulare ausgehend vom Ergebnis gestalten, verhindern Sie fast schon automatisch unnötige Abfragen. Zunächst analysieren Sie das tatsächliche Ziel des Formulars, dann werden die zu Erreichung dieses Ziels unbedingt notwendigen Daten festgelegt, die zur Verarbeitung der Eingaben zwingend erforderlich sind (also üblicherweise der letzte Schritt in einem Formularprozess). Von diesen Daten ausgehend werden nun die Bestandteile des Formulars quasi »von unten nach oben« festgelegt. Das Ergebnis dieses umgekehrten Ansatzes ist meist ein wesentlich schlankeres Formular als bei der üblichen Methode, zunächst einmal alles in eine Anwendung hineinzustopfen, was im schlimmsten Fall ein Komitee oder Formular-Arbeitskreis als unverzichtbare Features festgelegt hat.

Bei dieser Analyse sollten Sie beachten, dass Prozesse in Formular-Anwendungen aus mehreren, in der Regel vier Ebenen aufgebaut sind:

1. Die Anordnung der Abfragen im Formular (das Layout, Hierarchien und Abstände, die Typografie, aber auch sekundäre Elemente wie Fortschrittsanzeigen und Hilfen)
2. Die Fragen und die dazu gehörenden Antworten in ihren verschiedenen Formaten (die kleinsten Einheiten eines Formulars, mit denen einzelne Daten eingesammelt werden)
3. Die Zusammenhänge zwischen den einzelnen Abfragen im Formular (inkl. eventueller Verzweigungen, die üblicherweise in einem Diagramm dargestellt werden)
4. Die Bearbeitung sowie weitere Aktivitäten und Inhalte um das Formular herum (der Workflow selbst, aber auch die Pfade über die Nutzer zur Formularanwendung gelangen).

Dabei bietet sich an, das Design (Design wird von uns im allumfassenden Sinne verwendet, nicht als pure Dekoration der Oberfläche) beim letzten Punkt zu beginnen, weil dieser Punkt die folgenden beeinflussen wird. Bereits zu diesem Zeitpunkt sollten möglichst alle Beteiligten (die Nutzer der Formulare und die Nutzer der eingegebenen Daten im Unternehmen) in diesen Prozess einbezogen werden – nur so können Sie frühzeitig herausfinden, ob die unterschiedlichen Ansprüche an das Formular miteinander vereinbar sind und eventuelle Differenzen bereits auflösen, bevor es beim fertigen Produkt zu spät ist.

Erst nachdem Sie den minimal erforderlichen Satz an Abfragen festgelegt haben sollten Sie sich an die weiteren Punkte machen, denn Änderungen bei den grundlegenden Dingen verursachen unnötige Doppel-Arbeiten, wenn gleichzeitig bereits am Layout einer Anwendung gearbeitet wird, die noch nicht fertig konzipiert ist.

Bei der Entwicklung von web-basierten Formularen werden allzu oft die Vorgaben aus einem Pflichtenheft oder sogar vorhandene Papierformulare 1:1 in Formularfelder und ihre Beschriftungen übersetzt. Dabei sollte man sich zunächst einmal die Abfolge der Abfragen intensiv anschauen. Dann legen Sie fest, welche Abfragen bzw. Antworten jetzt nötig sind oder auf später verschoben werden können. Jede Abfrage muss der Nutzer aufnehmen und verarbeiten, er muss eine passende Antwort formulieren und diese in das entsprechende Feld eingeben. Jedes Feld das Sie einsparen, erspart ihren Nutzern Arbeit. Das Ergebnis: Ihr Formular kann schneller bearbeitet werden und die Wahrscheinlichkeit wächst, dass der Formularprozess vollständig durchgeführt wird.

Nutzer brechen eine Transaktion bei missverständlich formulierten Abfragen oft ab oder wenn sie sich fragen»Warum wollen die denn ausgerechnet das jetzt von mir wissen?« Die Reise-Site Expedia hat durch Beobachtung von Benutzern festgestellt, dass viele Nutzer ein (zudem noch überflüssiges) Eingabefeld falsch interpretierten, was in Folge zu einer Fehlermeldung des Systems führte. Nachdem das missverständliche Feld kurzerhand entfernt wurde, konnte ein Umsatz-Plus von 12 Millionen US$ gemessen werden.

Weitere entscheidende Faktoren für den Erfolg eines Formulars sind die Abfolge der Schritte und die Anordnung der Abfragen zueinander. Eine Reihe von Abfragen kann oft besser in Kategorien unterteilt werden kann, die dann getrennt voneinander präsentiert werden. Ebenso wichtig wie die Abfolge ist, dass dem Nutzer, wie Luke Wroblewski es nennt,

Ein indirekter Nebeneffekt der Reduzierung auf das Nötigste ist der, dass Ihre Anwendung dadurch automatisch performanter wird. Große Websites wie Google oder Amazon beobachten sehr genau die Zeiten, die ihre Websites zum La-

ein »clear path to completion« gezeigt wird, also ein deutlicher Pfad zur Vervollständigung der nötigen Eingaben: Von den Eingabefeldern und Kontrollelementen bis hin zum Element, dass die Eingaben abschickt, muss für den Nutzer klar ersichtlich sein, was von ihm verlangt wird und was der nächste Schritt ist. Der Nutzer muss möglichst effizient und zufriedenstellend das Ziel erreichen, dass er mit dem Ausfüllen des Formulars verfolgt: ein getätigter Einkauf, eine erfolgreiche Registrierung oder das Einstellen von Inhalten. Formulare, die Eingaben lose auf einer Seite verteilen und keine logischen Abläufe erkennen lassen, beeinträchtigen die Fähigkeit des Nutzers, passende Antworten auf die Fragen zu finden und den Prozess zu absolvieren.

den benötigen. Sie stellen dabei immer wieder fest, dass bereits kleine Verzögerungen von wenigen hundert Millisekunden teilweise negative Auswirkungen auf den Traffic im zweistelligen Bereich haben. So sprach Marissa Mayer, Vice President von Google, auf der Web2.0-Konferenz im Jahre 2006 davon, dass nur eine halbe Sekunde Verzögerung im Aufbau der Seiten einen 20%igen Einbruch im Traffic nach sich zogen.

Zur umfassenden Information des Nutzers gehört eine Anzeige, wie viele Schritte in einem mehrteiligen Formularprozess bereits bewältigt sind und wie viele dem Nutzer noch bevorstehen. Hier ein Screenshot einer solchen Fortschrittsanzeige aus der Anmeldung zum BIENE-Wettbewerb 2010:


## Notwendige Komplexität

Daher ist der gegenwärtige Trend zur Reduzierung und Vereinfachung gerade aus Sicht der von Barrieren Betroffenen zu begrüßen – ehrlicherweise allerdings nur bis zu einem gewissen Punkt. Das Leben ist komplex, und unsere Werkzeuge müssen in gewisser Weise diese Komplexität abbilden können. Auch scheinbar simple Dinge können verwirrend sein, genauso wie es komplexe Dinge gibt, die sehr gut verständlich sind. Es geht also nicht darum, Features durch Wegnehmen zu reduzieren, sondern die notwendigen Features so zu gestalten, dass sie von den Anwendern genutzt werden können – erst das ist gutes Design, und das Design von Einfachheit gehört bekanntlich zu den schwierigsten Aufgaben dieser Disziplin.

In der Summe lässt sich die Komplexität einer Formular-Anwendung nicht reduzieren, Sie können nur die Komplexität auf andere Schultern verlagern, z.B. vom Frontend, das der Benutzer sieht zum Backend, an dem der Administrator sitzt.

## Unnötige Komplexität

Gerade bei Formularen oder web-basierten Anwendungen gilt das alte Netz-Mantra, dass man bei dem, was man ausliefert, strikt sein sollte und bei dem, was man akzeptiert, tolerant sein sollte. Diese natürliche Fehlertoleranz gegenüber dem Nutzer nimmt einer Anwendung schon viel an möglicher Komplexität, ist aber leider viel zu selten zu beobachten. Dabei würden einige kurze Tests mit echten Nutzern aufzeigen, was diese falsch machen können *(und werden!)* und damit Hinweise geben, wie man diese Fehler auf Seiten des Anbieters am besten von vornherein verhindert.

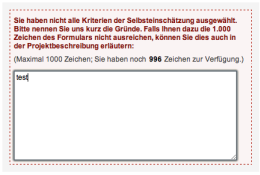
Ein Beispiel für mangelnde Fehlertoleranz ist der Zwang, Eingaben so zu tätigen, wie es der Anbieter gerne hätte. Die dahinter stehenden Anforderungen oder Konventionen müssen dem Nutzer jedoch nicht unbedingt bekannt sein. So sollten Sie bei Eingabefeldern für Telefonnummern lediglich ein Feld anbieten und in diesem alle Schreibweisen akzeptieren, statt es auf zwei (Vorwahl/Durchwahl) oder sogar drei (mit Ländervorwahl) aufzuteilen. Unterschiedliche Eingaben (`+49 – 22 8 – 20 92 0` oder `022820920`) können Sie serverseitig auflösen, zumal viele dieser Schreibweisen durch Normen wie die DIN 5008 sogar standardisiert sind. Eine Untersuchung bei einem australischen Mobilfunk-Anbieter hat ergeben, dass es bis zu 40 Varianten gibt, in denen Nutzer ihre Telefonnummer eingeben, wenn man ihnen die Möglichkeit dazu lässt. Dabei müssen Sie noch nicht mal besondere Hinweise auf die Möglichkeit der freien Eingabe hinterlegen – Nutzer tendieren bei fehlenden Hinweisen eher dazu, Daten in der für Sie gewohnten Form einfach einzugeben.

Eine weitere Unterteilung in mehrere Textfelder wäre hier in der Regel nicht barrierefrei zu nutzen, da z.B. ein Label immer nur für ein einziges Formularelement definiert werden kann

und somit eine logische Verknüpfung mit weiteren Feldern nicht möglich ist. Besonders für blinde Nutzer ist dies ein Problem – sie werden versuchen, die kompletten Daten im ersten Feld einzugeben und regelmäßig an dieser Stelle scheitern. Bei kalendarischen Daten hat sich mittlerweile durchgesetzt, zusätzlich zu den Eingabe- oder Auswahlfelder für Tag/Monat/Jahr auch noch einen echten Kalender in Form eines so genannten Date-Pickers anzubieten. Dies hat den Vorteil, dass hiermit in einem einzigen Kontrollelement umfassendere tabellarische Daten (Kalendertage, Kalenderwochen, Wochenenden etc.) angezeigt werden können, womit man dem Nutzer eine Menge Denkarbeit abnimmt.

Gleiches gilt für alle anderen Zahlen- oder Textfelder, in denen der Nutzer freie Eingaben machen kann, wie Kreditkartennummern, Datumsangaben etc. Auch hier sollten Sie Eingaben mit und ohne Leerschritte, mit oder ohne Zeichen wie / für die Trennung in Telefonnummern oder den durchaus üblichen Punkt als Trennung für Tausender akzeptieren. Besonders fatal auf die Kundenzufriedenheit dürften sich Abfragen auswirken, bei denen das System selbst Daten in einer bestimmten Schreibweise ausgibt, diesen Vorschlag dann aber wie im folgenden Screenshot als Eingabe nicht mehr akzeptiert.


Werden Daten zwingend in einem bestimmten Format oder in einer festgelegten Länge benötigt, dann sollten Sie dem Nutzer vor der Eingabe erklären, was zulässige Eingaben sind. Diese Eingaben validieren Sie dann in einem weiteren Schritt. Bei einer natürlichen, in der Art der Sache liegenden Begrenzung der Eingabe (z.B. maximal 160 Zeichen beim SMS-Versand oder 140 Zeichen bei Twitter) können und sollten Sie sogar einen Zähler anbieten, der die verbleibenden Anschläge anzeigt. Generell gilt die Regel bei solchen Begrenzungen: was im `maxlength`-Attribut steht, sollte auch noch mal im Klartext daneben stehen (idealerweise im Label des betreffenden Feldes, wie der folgende Screenshot aus der Einreichung zum BIENE-Wettbewerb zeigt):




Wenn Sie die Länge von Eingaben begrenzen, dann sollte sich Ihr HTML-Code auch daran halten und nicht stattdessen Fehler provozieren. Beispielsweise Passwort-Felder mit der Beschriftung, dass ein Passwort nur 16 Zeichen lang sein darf, die dann zunächst die Eingabe von mehr als 16 Zeichen zuließen, nur um danach eine entsprechende Fehlermeldung auszugeben.

Um beim obigen Kalender-Beispiel zu bleiben: hier kann es durchaus sinnvoll sein, nicht mehr wählbare (weil z.B. in der Vergangenheit liegende) Daten kurzerhand nicht mehr anzuzeigen oder zumindest optisch und technisch zu deaktivieren. Wenn in einem Produkt-Konfigurator bestimmte Optionen nicht miteinander kombinierbar sind, dann sollten Sie den Nutzer nicht dazu verleiten, diese »Fehl«-Konfiguration zu tätigen, nur um ihm hinterher eine Fehlermeldung zu präsentieren. Was nicht geht hat im Formular oder in der Anwendung nichts zu suchen.

Beim Vergleich verschiedener Systeme, z.B. im e-Commerce-Bereich, stößt man immer wieder auf Umsetzungen, die zwar formal barrierefrei sind und alle technischen Kriterien erfüllen, die aber so komplex strukturiert sind, dass sie kein Nutzer mehr (trotz der technischen Zugänglichkeit) versteht. Einen interessanten Vergleich fanden wir in der Analyse zweier im Wettbewerb zueinander stehender Finanz-Programme. Mark Hedlund, der Entwickler eines Konkurrenz-Produkts schreibt hier:

> *»I was focused on trying to make the usability of editing data as easy and functional as it could be; Mint was focused on making it so you never had to do that at all. Their approach completely kicked our approach's ass.«*

Ein weiteres Beispiel für unnötige Komplexität sind die verbreiteten Funktionen zur »erweiterten Suche«, die viele Websites anbieten. Es muss nicht immer ein solcher Extremfall wie in diesem Beispiel sein (Screenshot), und es gibt sicher berechtigte Ausnahmen wie z.B. umfangreiche Bibliotheks-Recherchen, aber im Regelfall gilt für die allermeisten Content-Sites: wenn ein Nutzer die einfache Suche nicht versteht, hilft ihm die erweiterte Suche meist auch nicht weiter.

Bei der Abfrage von Adressen können Sie bereits vorhandene mentale Modelle der Nutzer berücksichtigen und da-

mit mögliche Komplexität entschärfen. Durch jahrelanges Training im praktischen Umgang haben Nutzer eingebrannte Vorstellungen davon, wie eine Adresse aufgebaut ist. Generell gehen Adressen vom Spezifischen (Name zuerst) zum Allgemeinen (Land zuletzt), die internationalen Unterschiede liegen meist nur in der Anordnung der Daten zwischen diesen beiden Polen. Genau diese machen aber einen Adressblock erst auf den ersten Blick erkennbar: wenn die Kombination aus Straße/Hausnummer/Postleitzahl/Ort entsprechend den Sehgewohnheiten präsentiert wird, ist der Adressblock und damit die Art der einzugebenden Daten (zumindest für sehende Nutzer) erkennbar, auch ohne dass sie die Feldbeschriftungen gelesen haben.

Wenn Sie jedoch mehrsprachige Inhalte und Funktionen anbieten, dann sollten Sie Ihren Besuchern mit Adressen in anderen Ländern den gleichen Komfort bieten, sodass auch diese auf einen Blick erkennen können, wo der CAP, Code postal, ZIP Code oder Postcode einzugeben ist. Hierzu müssen Sie unter Umständen Anordnung und Größe der Felder anpassen, dieser einmalige zusätzliche Aufwand wird durch höhere Komplettierungsraten wettgemacht.

Weitere Tips zur Berücksichtigung internationaler Adress-Formate finden Sie im Artikel »International Address Fields in Web Forms«

Eine unschöne Marotte von Formularen ist die doppelte Abfrage von E-Mail-Adressen, z.B. bei Registrierungen. Das vermeintliche Ziel ist klar: damit soll die Wahrscheinlichkeit einer Fehleingabe halbiert werden, da man ansonsten die Gültigkeit einer E-Mail-Adresse nur durch eine Mail an ebendiese verifizieren kann. Das Problem mit dieser Methode:

- es unterstellt dem Nutzer von vornherein, einen Fehler zu machen,
- es zwingt ihn zu mühsamen Mehrfacheingaben,
- es löst das Problem falscher Eingaben nicht , da die meisten Nutzer einfach die einmal eingegebene E-Mail-Adresse aus dem ersten Feld in das zweite Feld kopieren (und damit eventuelle Fehler gleich mit).

Mittlerweile beherrschen alle modernen Browser das Ausfüllen von Formularen mit Daten aus dem lokalen Adressbuch des Rechners, was den Zwang zum doppelten Ausfüllen ad absurdum führt.

## Nicht so eilig!

Für Nutzer mit kognitiven Behinderungen stellen Zeitbegrenzungen und ablaufende Sitzungen eine echte Hürde dar. Diese Nutzer haben oftmals nicht die nötige Ausdauer, um komplizierte Abläufe an einem Stück abzuarbeiten oder sie schaffen es nicht in der vorgegebenen Zeit. Bei komplexen oder mehrstufigen Formularen sollten der Nutzern die Möglichkeit haben, Prozesse zu unterbrechen, Zwischenstände zu speichern und zu einem späteren Zeitpunkt wieder aufzunehmen. Meist handelt es sich typischerweise um Anwendungen, bei denen der Nutzer bereits ein Nutzerkonto besitzt und somit identifizierbar ist, eine Nachverfolgung ist also kein Problem.


Zum Abschluss eines Formularprozesses muss der Nutzer seine Eingaben kontrollieren können. Insbesondere bei Vorgängen mit erheblicher Tragweite soll erst nach der Kontrolle der Eingaben die eigentliche Übernahme der Formulardaten auf den Server geschehen, um den Vorgang abzuschließen. Bei Änderungen an vorhandenen Datensätzen sollten Sie die nun manipulierte Ursprungsseite nochmals anzeigen, evtl. mit optischen Indikatoren und/oder einer textlichen Zusammenfassung der Änderungen.

## Fehlerbehandlung in Formularen

Viele Fehler lassen sich schon von vornherein verhindern, wenn man dem Nutzer die Möglichkeit gibt, die Daten so einzugeben wie er es gewohnt ist. Statt in der client- und serverseitigen Logik der Anwendung auf eine bestimmte Form der Eingaben zu bestehen, werden die Daten dann in einem zweiten Schritt auf dem Server normalisiert (d.h. für die Weiterverarbeitung z.B. einer Bestellung in eine gemeinsame Schreibweise gebracht).

Trotz aller Vorkehrungen lässt sich jedoch nie ganz verhindern, dass Nutzer wirkliche, echte Fehler machen – die Frage ist nur, wie Sie als Anbieter damit umgehen? Sinnvolle und nachvollziehbare Fehlermeldungen können nicht nur Verwirrung bei den Nutzern verhindern, sondern vermeiden auch kostspielige Anfragen beim Kundendienst. Wie

Chrissie Brodigan im Artikel »10 Tips on Writing Hero-worthy Error Messages« schreibt: »Error messaging is customer support«.

Lustige Bilder sind kein Ersatz für Fehlerbeschreibungen in Textform, die durchdacht und sinnbildend geschrieben sein müssen, um dem Nutzer wirklich weiter zu helfen. Bei Formularen und insbesondere bei Anleitungen und Fehlermeldungen greifen die Regeln der WCAG bzw. der BITV zum allgemeinen Verständnis und zur einfachen Sprache . So sollten Fehlermeldungen wie die Folgende eigentlich der Vergangenheit angehören:

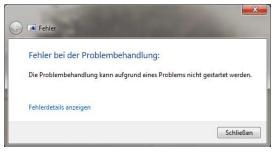

Wir treffen oft auf Situationen, in denen wir ein langes Formular ausgefüllt und abgeschickt haben, nur um anschließend vor einem leeren Formular mit lauter rot markierten Feldern zu sitzen – eine Seite die einem förmlich *»FAIL!«* entgegenschreit! Zu den unverzeihlichsten Fehlern, die eine Anwendung begehen kann gehört, dass alle Daten verloren gehen, wenn der Nutzer auch nur ein Feld falsch bedient hat. Wenn Sie nicht gerade in einem Bereich mit extremen Ansprüchen an Datenschutz und -sicherheit operieren, dann sollten einmal korrekt eingegebene Daten (natürlich mit der Ausnahme von Passworten) erhalten bleiben, sonst brichtder Nutzer an dieser Stelle die Transaktion frustriert ab.

Die Daten über Stellen im Prozess, an denen Nutzer abbrechen, besitzen Sie eventuell bereits: die Logfiles des Servers, deren Analyse die Schwachpunkte zeigt, an denen Nutzer aufgeben. Untersuchen Sie diese Daten bezüglich der Fragen:

- Welche Felder werden falsch ausgefüllt?
- Was haben Nutzer in diese Felder eingegeben (oder vergessen einzugeben)?
- Was sind die häufigsten Fehlermeldungen?
- Wie viele (Fehl-)Versuche benötigen die Nutzer?
- Wie hoch ist die Abbrecher-Quote (und an welcher Stelle wird abgebrochen)?

# Verständliche Hilfen

Schon in der Konzeptionsphase sollten Sie sich über die Art der Rückmeldung auf Eingabefehler Gedanken machen und diese auch testen, egal ob Sie mit Wireframes oder Prototypen arbeiten. Diese Überlegungen haben erhebliche Auswirkungen auf die technische Umsetzung. **Dass** Sie Ihre Nutzer über Fehler informieren sollten, versteht sich von selbst. **Wie** Sie Ihre Nutzer über Fehler informieren ist eine Frage der Philosophie und des Kontextes innerhalb der Anwendung. Die einzige Regel, die immer und überall gilt: wenn ein Fehler passiert, dann sollten Sie den Nutzer möglichst schnell und möglichst »laut« benachrichtigen.




Quelle: laborenz.de/lab-o-log

Wann und in welcher Intensität dies geschieht ist Gegenstand langer Debatten unter Webdesignern und spiegelt die unterschiedlichen Ansätze der verschiedenen Betriebssysteme wieder. Typische Windows-Anwendungen konfrontieren den Nutzer eher mit zu vielen Meldungen (»Es ist nichts passiert. Bitte drücken Sie OK oder Abbrechen, um keine Änderungen zu bestätigen«), Betriebssysteme aus der Unix-Welt wie Linux, BSD oder Mac OS X hingegen sagen nichts, wenn nichts bemerkenswertes passiert ist (geben also in der Regel auch keine Bestätigungen, wenn ein Vorgang ohne Fehler abgelaufen ist).

Die beste Lösung ist der goldene Mittelweg zwischen »zu viel« und »zu wenig« – wo dieser liegt müssen Sie im Kontext Ihrer Anwendung selbst herausfinden und durch Nutzertests verifizieren. Ein typisches Beispiel von »zu viel«, das leider viel zu oft im Netz zu finden ist, sehen Sie im folgenden Screenshot:


Hier kann der Nutzer den Alert nur per OK wegklicken, wird dann aber nicht zu den fehlerhaften Stellen geführt. Wenn Ihr Formular nur aus einer geringen Anzahl Felder besteht und somit die Fehleingaben überschaubar sind, weisen Sie den Nutzer besser einzeln auf die Fehler hin, wie der folgende Screenshot zeigt:


Mit JavaScript (z.B. `document.forms["bestellung"].vorname.select();`) wird dann beim Bestätigen der Warnmeldung per OK-Button der Fokus direkt auf das jeweilige Eingabefeld gesetzt. Diese Methode hat ihre Grenzen: ab einer gewissen Anzahl Felder (und damit potenzieller Fehler) ist es sinnvoller, diese dem Nutzer nicht als endlose Folge von einzelnen Warnmeldungen zu präsentieren, sondern sie als Auflistung im Formular selbst anzuzeigen:

```
<form id="bestellung">
…
  <h4>Fehler!</h4>
  <p>Folgende Felder müssen ausgefüllt werden, um Ihre Bestellung abzuschließen:</p>
  <ol>
        <li><a href="#label-vorname">Bitte geben Sie Ihren Vornamen ein</a></li>
        <li><a href="#label-nachname">Bitte geben Sie Ihren Nachnamen ein</a></li>
        <li><a href="#label-strasse">Bitte geben Sie Ihre Straße ein</a></li>
  </ol>
  …
  <label for="vorname" id="label-vorname">Vorname:</label>
  <input type="text" id="vorname">
…
```

Bei einem Reload der Fehlerseite ändern Sie zusätzlich noch den `TITLE` der Seite sowie deren primäre Überschrift. Hier sollte der klare Hinweis stehen, dass es sich um eine Seite mit Fehlerhinweisen und nicht um die Bestätigungsseite handelt (unter anderem weil der Titel einer Seite im Screenreader immer zuerst vorgelesen wird).

Wenn Sie Hinweise, Hilfen oder Fehlermeldungen ausgeben, sollten Sie dies unter keinen Umständen in Form eines der in letzter Zeit in Mode gekommenen Overlays nach der Art einer Lightbox tun. Diese Art der Präsentation von Dialogen bringt keinen zusätzlichen Nutzen gegenüber echten Dialogen oder Inline-Hinweisen. Im Gegenteil: es treten weitere Probleme auf, man kann mit der auslösenden Seite nicht mehr interagieren, ohne dass die Hinweise verschwinden, denn jeder Klick ausserhalb des Overlays (also auch zur Fehlerkorrektur) schliesst diesen. Dies ist bei mehreren Fehlermeldungen in diesem Overlay besonders schwierig, da der Nutzer gezwungen wird sich sämtliche Meldungen zu merken weil mit dem Schließen die Fehlermeldungen verschwinden.

## Wahrnehmbare Hilfen

Auch bei deutlichen Hinweisen auf Pflichtfelder, akzeptierte Daten und Formate, kann und wird es passieren, dass diese Hinweise übersehen werden. Dann muss Ihre clientseitige und die serverseitige Programmierung in der Lage sein, das Formular entsprechend verändert zurückzugeben und nur die fehlenden Eingaben erneut abzufragen.

Dazu gehört auch ein deutlicher Hinweis, welches die fehlerhaften Felder sind. Das kann in Form einer Hervorhebung durch Text, Form und Farbe geschehen. Ein einfaches Umfärben des Textes z.B. in die Signalfarbe rot reicht nicht aus, da 8-10% aller Männer farbfehlsichtig sind und diese Hilfe nicht erkennen.

Verbreitete Screenreader lesen nicht alles vor, was in einem Formular innerhalb des <form>-Elements steht. Screenreader verwenden einen sogenannten Formularmodus um die für die Bearbeitung von Formularen benötigten erweiterten Tastaturkürzel zur Verfügung zu stellen. Dieser unterscheidet sich von den herkömmlichen Navigations- oder Vorlesemodi dadurch, dass in Formularen nur die Inhalte der Formularelemente (`<legend>, <input>, <textarea>` etc.), ihre Beschriftungen (`><label>`) sowie Links (`<a href="…">`) vorgelesen werden. Sämtliche anderen Inhalte wie Texte oder Überschriften, die nicht in den klassischen Formularelementen stehen, werden ignoriert. Eine falsche Strukturierung führt dann dazu, dass Hilfen oder konkrete Fragen in einem Formular nicht wahrgenommen werden. So liest ein typischer Screenreader im folgenden Beispiel nur die Label »Ja« und »Nein« vor – die eigentliche Frage bleibt dem Nutzer verborgen:

```
<form>
    <p>Möchten Sie Sardellen auf Ihrer Pizza?</p>
    <input type="radio" name="belag" id="ja" value="anchovis-ja" checked="checked">
```

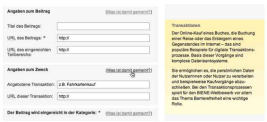
```
<label for="ja">Ja</label>
<input type="radio" name="belag" id="nein" value="anchovis-nein">
<label for="nein">Nein</label>
```

Hier ein separates Fieldset einzuführen und die Frage als Legende (`<legend>Möchten Sie Sardellen auf Ihrer Pizza?</legend>`) zu hinterlegen, würde ein Mehr an Code und damit ein Mehr an Komplexität bedeuten, die an dieser Stelle unnötig ist. Eine mögliche Lösung für dieses »Problem«: wenn Ihr Formular ohne diese Texte Gefahr läuft, nicht verstanden zu werden, dann können Sie die benötigten Texte in Links setzen und diese mit der ID des Labels verknüpfen.

Die Antwort auf die Frage nicht als zwei Radiobuttons, sondern als eine Checkbox und die Frage als deren Label zu hinterlegen ist also die beste Lösung Eine Checkbox erfüllt hier aufgrund des binären Charakters der Antwort (Haken = Ja, kein Haken = Nein) denselben Zweck wie zwei Radiobuttons.

Barrieren entstehen auch durch die mangelhafte Kennzeichnung der Fehleingaben. Die Hinweise werden meist nicht in unmittelbarem Zusammenhang mit dem betreffenden Kontrollelement gezeigt, sondern weit entfernt davon und nicht sinnvoll verknüpft. Gerade bei nicht-grafischen Zugangsarten wird hierdurch eine Zuordnung schwierig. Oft ist es für den Nutzer einfacher, die gesamte Prozedur von vorne zu beginnen, wenn die Fehlermeldungen nicht sinnvoll mit dem Ort des Fehlers verknüpft sind.

Die oben genannte Technik der lokalen Sprungmarken können Sie sich zu Nutze machen, um zusätzliche Hinweise und Erklärungen ( z.B. Hilfen oder Informationen zu Datenschutz und Widerrufsrecht) innerhalb eines Formulars zu verlinken. Im folgenden Screenshot sehen Sie einen Ausschnitt aus dem Anmeldeformular zur BIENE 2010, wo eventuell schwer verständliche Konzepte wie hier die Erklärung, was mit »Transaktionen« gemeint ist, direkt von den entsprechenden Eingabefeldern aus verlinkt sind. Zusätzlich wird der angesprungene Block mit der Erklärung durch die CSS-Pseudoklasse `:target` farblich hervorgehoben – hierzu später mehr im CSS-Kapitel.



Was wäre Webentwicklung ohne die passenden Browser-Bugs? Richtig: Langweilig. Hier gibt es wieder eine Reihe von Inkonsistenzen und abweichenden Implementierungen in den verschiedenen Browsern. Formularelemente, deren ID von einem Link aus angesprungen wurde, erhalten zwar den :target-Zustand und sind darüber per CSS formatierbar (z.B. mit einer zusätzlichen outline oder border zur besseren Hervorhebung); ob der Fokus aber ebenfalls auf das Formularelement gelegt wird, ist vom verwendeten Browser abhängig. Im Internet Explorer wird der Fokus korrekt auf das angesprungene Formularelement gelegt, dafür versteht dieser in der Version 6 & 7 die :target-Pseudoklasse nicht. Letztere wird von moderneren Browsern zwar verstanden, dafür behalten Safari und Chrome den Fokus auf dem ursprünglichen Link; Firefox hingegen verliert den Fokus komplett, d.h. weder der angeklickte Link noch das angesprungene Formularelement haben den Fokus.

## Bedienbare Hilfen

Während die Vorgaben zur Verständlichkeit in Richtlinien wie den WCAG oder Verordnungen wie der BITV uneingeschränkt anwendbar sind, können Sie andere Regeln ruhig etwas liberaler auslegen: so darf man den Sinn des recht pauschalen Verbots von Pop-Ups durchaus hinterfragen, gerade wenn es um Webangebote mit Anwendungscharakter geht. Nach strikter Lesart würden hierunter auch Fehlermeldungen (also echte Alerts) fallen, die jedoch bei korrekter Umsetzung auch in assistierenden Programmen gut nutzbar sind und geeignet sind, den Nutzer vor groben Fehlern zu bewahren.

Ähnliches gilt für Hilfe-Funktionen innerhalb einer Anwendung oder eines komplexen Ablaufs: hier kann es durchaus sinnvoll sein, diese in einem separaten (Pop-Up-) Fenster zu öffnen, damit der Nutzer im Bedarfsfall beides im direkten Zugriff hat. Auch diese stellen keine Barriere dar, sondern helfen dabei, diese zu beseitigen. Ob diese Pop-Ups angekündigt werden müssen ist Gegenstand langer Debatten, in die wir uns nicht einmischen möchten (zumal viele Arten von Pop-Ups aufgrund ihrer Natur gar nicht angekündigt werden können).

## Übertreiben Sie es nicht mit den Hilfen

Die Formulierung und der Umfang von Hilfen bedeutet auch immer eine Balance auf dem schmalen Grat zwischen hilfreich oder fördernd und herablassend oder sogar bevormundend – im Englischen wird hierfür das wesentlich besser passende Wort »patronizing« verwendet, für das es leider keine hundertprozentige deutsche Entsprechung gibt. Die Erfahrung aus Nutzertests hat jedoch gezeigt, dass die Wahrscheinlichkeit, dass Nutzer Hilfen wirklich lesen, umgekehrt proportional zum Umfang der Hilfen ist. Zu umfangreiche Unterstützung der Nutzer kann auch sehr schnell negative Konsequenzen auf deren Performance haben.

So berichtete Christof van Nimwegen in seiner Dissertation »The paradox of the guided user: assistance can be counter-effective (2008)« von einem Experiment, bei dem zwei verschiedene Systeme und die Leistung der jeweiligen Nutzer gegeneinander getestet wurden. Das eine System versuchte die Nutzer möglichst weitgehend zu unterstützen, indem möglichst viele triviale, mechanische Bestandteile der Aufgabe vom System übernommen wurden. Nutzer hatten also mehr Zeit, sich mit der eigentlichen Aufgabe zu beschäftigen. Das andere System wiederum stellte dieses Maß an Unterstützung nicht zu Verfügung, zwang also die Nutzer, sich mehr mit den inneren Details der Anwendung zu beschäftigen:

> *»One group of interfaces is more user-friendly, in that it attempts to aid the user as much as possible in their given task, for example, indicating options available at a given moment of their task (e.g. greying out and disabling unavailable options, or highlighting available moves). This interface tries to offload as much ›trivial‹ mechanical thinking as possible to the machine to give the user more time to think about the problem itself. The other group of interfaces don't provide this level of help and hints, forcing the user to get better acquainted with the mechanics of each problem and put more thought into how they will want to complete their task.*
>
> *In essence, one set of interfaces externalizes information, whereas the other internalizes it, in relation to the user. [...] the user-friendly interface relieves its users from having to commit information to memory, and this information is in turn externalized. When the interface isn't helpful, information is internalized by the user, meaning they have to think longer about the problem and learn more fully the inner workings of their task.«*

Entgegen alle Regeln und Mantras der Usability stellte sich heraus, dass die zweite Gruppe wesentlich besser bei der Bewältigung der Aufgaben abschnitt, weniger abgelenkt und konzentrierter bei der Arbeit war und zudem das erworbene Wissen auch auf andere Systeme transferieren konnte:

> *»The findings are very interesting and go against the general trend of simply accepting usability of user interfaces as something completely beneficial to the user. Indeed, the opposite thing happened. People using the more difficult interfaces tended to perform better, were less fazed by distractions and were found more likely to transfer their skills to new interfaces or tasks.*
>
> *People using the user-friendly interfaces tended to rely on them too much, and so were never fully able to grasp the problem and come up with working strategies for tackling it. Even though the user-friendly interface was meant to relieve them of trivial tasks, they instead relied on the computer as a crutch, stumbling around on their way to an eventual solution with the help of the interface.«*
>
> (zitiert nach: »The Dark Side Of Usability«)

## Ansprechende Texte

Zum Design der Abfragen, Bestätigungen und Fehlermeldungen gehört auch die Festlegung von Formulierungen, die dem Nutzer den Umgang mit der Anwendung erleichtern. Im Artikel »Usability Tip: Use Verbs as Labels on Buttons « spricht Dmitry Fadeyev davon, Verben mit Aufforderungscharakter statt abstrakter bzw. generischer Begriffe zu verwenden – eine Idee die sich nicht nur auf Dialoge beschränkt, sondern auch auf Feldbeschriftungen, Aufforderungen, Hilfetexte usw. übertragen lässt.

Wir alle kennen die üblichen Buttons in Dialogen: kurze Phrasen die uns auffordern, eine Aktion zu bestätigen oder abzubrechen. Im folgenden Screenshot sehen Sie einen solchen Dialog aus einer Textverarbeitung unter Windows, der angezeigt wird, wenn man ein Dokument schließt ohne es gespeichert zu haben:


Der Dialog fragt, ob man die Änderungen im Dokument speichern möchte. Danach folgen die Buttons »Ja«, »Nein« und »Abbrechen«. Deren Funktion ist aber erst offensichtlich, nachdem man den vorstehenden Text ebenfalls gelesen hat; für sich genommen geht ihr Informationswert gegen Null.

Der nächste Screenshot ist aus einem vergleichbaren Programm unter MacOS X:


Die Nachricht ist ähnlich wie unter Windows, nur geht sie mehr ins Detail und erklärt was passiert, wenn man die Änderungen nicht sichert. Der interessante Unterschied liegt aber in der Betextung der Buttons: hier finden sich nur Verben (»Nicht sichern«, »Abbrechen« und »Sichern«), die alle auch ohne ihren Kontext verständliche Aussagen über die dahinter liegende Aktion machen. Im ersten Beispiel muss man in der Tat die gesamte Dialogbox lesen, um die Aktion zu verstehen und eine Entscheidung zu treffen, im zweiten Beispiel kommt der Nutzer in kürzerer Zeit zu einer klareren Entscheidung.

### Eine kurze Checkliste

- Teilen Sie dem Nutzer mit, dass ein Fehler aufgetreten ist.
- Machen Sie deutlich was der Fehler ist und an welcher Stelle er aufgetreten ist.
- Geben sie dem Nutzer die Hilfen die er benötigt, um den Fehler zu beheben oder um einen Ausweg aus diesem Zustand zu finden.

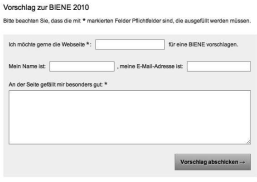
Eine ganze Sammlung hervorragend gemachter Fehlermeldungen finden Sie in der UI Patterns Library unter »Input Feedback«.

## Um die Ecke denken

Formulare sind von ihrer Natur her etwas sehr mechanisches – die Menschen, die sie ausfüllen sollen, hingegen eher nicht. Oft ist es für Nutzer einfacher, ihr Anliegen in Textform auszuformulieren, statt Daten aus einer nicht enden wollenden Anzahl von Optionen auszuwählen. Zumal diese Optionen oftmals gar nicht die gesamte Bandbreite der möglichen Eingaben abdecken können (wie oft standen Sie schon vor einem Formular, dass nur ein »Ja« oder ein »Nein« anbot, die aus Ihrer Sicht richtige Antwort wäre aber ein »Vielleicht« gewesen?). Warum also nicht dem Nutzer die Freiheit lassen, die Daten so einzugeben wie er es möchte? Das war die Überlegung hinter der Neugestaltung des Formulars für die Nutzer-Vorschläge beim BIENE-Wettbewerb 2010: dem Nutzer sollte das Formular in Form eines Lückentexts angeboten werden. Auszufüllen waren nur ein paar Lücken in den Sätzen:

> *»Ich möchte gerne die Webseite: [ ] für eine BIENE vorschlagen. Mein Name ist: [ ], meine E-Mail-Adresse ist: [ ]. An der Seite gefällt mir besonders gut: [ ].«*

Lediglich der URL des Vorschlags (ohne diesen geht es nicht) und eine Begründung, warum man gerade dieses Angebot gut findet, waren Pflichtfelder; der Rest optional.




Ob diese Umgestaltung tatsächlich zu der gemessenen Zunahme der Vorschläge geführt hat, können wir nicht wirklich nachweisen (dazu fehlt ein A/B-Test), aber zumindest gab es über dieses Formular 20% mehr Vorschläge als über das herkömmlich aufgebaute Formular des Vorjahres.

Diese Beobachtung deckt sich mit den Erkenntnissen bei anderen Sites, wo vergleichbare Formulare eingesetzt werden: Luke Wroblewski spricht von Zunahmen um 25-40% durch diesen nach einem Kinderspiel benannten *»Mad-Libs-Style«*. Der Grund scheint zu sein, dass Nutzer durch die Art der Abfrage und der Präsentation des Formulars nicht aus ihrem Aktivität herausgerissen werden; stattdessen reiht sich die Abfrage der Daten nahtlos in die Aktivität ein und unterstützt statt zu unterbrechen.

Dies ist übrigens auch der Grund, warum hier bei Einfach für Alle die Reihung der Formularfelder für Kommentare im Blog von der üblichen Reihenfolge (1. Vorname 2. Nachname 3. E-Mail 4. URL 5. Kommentar) abweicht. Aus-

gehend von der Überlegung, dass Nutzer ja schließlich primär einen Kommentar und nicht als erstes Ihren Namen abgeben wollen, wird folgerichtig zuerst das Kommentarfeld gezeigt, die sekundären Daten (Name, E-Mail, URL) folgen erst danach.


Die gesamte Serie auf einen Blick:

- Einleitung: Reine Formsache
- Konzeption: Toleranz und Rücksicht einplanen
- Formulardesign: der wichtige erste Eindruck
- Technik: HTML & CSS für Formulare
- Dynamik in Formularen: JavaScript & AJAX
- Testen von Formularen und Web-basierten Anwendungen
- Literatur
- Druckversion des kompletten Artikels

Glossar:

Erklärung der Fachbegriffe:
Browser Client CSS Screenreader Textauszeichnung Usability WCAG

Mehr dazu:

- Simon Willison: »Simple Tricks for More Usable Forms«
- Philipp Lenssen: »The Worst Form in the World«

Kommentare:

- Thomas Seidelmann zu: »Einfach für Alle – Digitale Barrierefreiheit für mobile Arbeits- und Lebenswelten«
- Boris Schneider zu: »Einfach für Alle – Digitale Barrierefreiheit für mobile Arbeits- und Lebenswelten«
- Hans Peter Sperber zu: »Screenreader nutzen – Anleitung für NVDA«
- Tina zu: »Einfach für Alle – Digitale Barrierefreiheit für mobile Arbeits- und Lebenswelten«
- Olaf Drümmer zu: »Techniken für barrierefreie PDFs auf Deutsch«