Die eigene Website

Zusatzkapitel „JavaScript für Fortgeschrittene“

Im Folgenden finden Sie einige Zusatztipps zu Kapitel 9 des Buchs „Die eigene Website“. Mehr Infos zum Buch auf der Website zum Buch.

Text auf der Seite ändern

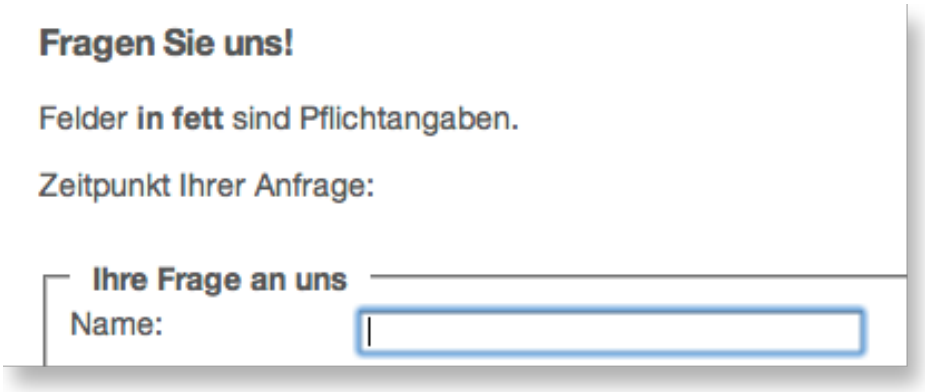
Sie können mit JavaScript auch direkt Text auf HTML-Seiten schreiben.

1. Ergänzen Sie auf der HTML-Seite des letzten Beispiels vor dem Formular:

```
<script type="text/javascript">document.write("<p>
Zeitpunkt Ihrer Anfrage: <p>")</script>
```

2. Speichern und testen Sie die Seite.

Denken Sie daran, dass innerhalb einer JavaScript-Anweisung keine Umbrüche vorkommen dürfen.



Der dritten Zeile sieht man im Browser nicht an, dass sie mit JavaScript geschrieben wurde.

Der Seite im Browser sehen Sie nicht an, dass der Text von JavaScript geschrieben wurde. Wir hätten das genauso gut noch als reinen HTML-Text schreiben können. Die folgenden Dinge gehen aber nur mit JavaScript:

Auch wenn es mit JavaScript geht: Verzichten Sie auf die Angabe von Datum und/oder Uhrzeit irgendwo auf Ihrer Website, wenn sie nicht wirklich sinnvoll ist. Um die Zeit zu erfahren, kommen die Besucher nicht zu Ihnen – es lenkt nur ab.

3. Ergänzen Sie den Code:

```
var meinZeitobjekt = new Date();
var Tag = meinZeitobjekt.getDate();
var Monat = meinZeitobjekt.getMonth() + 1; // Januar ist 0
var Jahr = meinZeitobjekt.getFullYear();
var Datum = Tag + "." + Monat + "." + Jahr;
document.write("<p>Zeitpunkt Ihrer Anfrage:
" + Datum + " Uhr<p>");
```

In der ersten Zeile haben Sie mit `new` ein so genanntes Objekt angelegt. Das `Date`-Objekt ist in JavaScript definiert und kennt einige Befehle. Diese nutzen wir in den nächsten Zeilen. `getDate` etwa liefert

© Matthias Gidda, Jens Jacobsen

den aktuellen Tag, `getMonth` den aktuellen Monat usw. Nachdem die Monatszählung bei 0 beginnt, addieren wir 1. Das ist wieder unpraktisch, aber das müssen wir hinnehmen – wenn man es weiß, ist es kein Problem.

Die Seite mit dem ganzen Code finden Sie hier: http://website.benutzerfreun.de/javascript/JS-Formular-4.html

4. Speichern und testen Sie.

Hier sehen Sie den gesamten Code des Skripts, noch ergänzt um die Angabe der Uhrzeit:

```
<script type="text/javascript">
    var meinZeitobjekt = new Date();
    var Tag = meinZeitobjekt.getDate();
    var Monat = meinZeitobjekt.getMonth() + 1; // Januar ist der Monat 0
    var Jahr = meinZeitobjekt.getFullYear();
    var Datum = Tag + "." + Monat + "." + Jahr;
    var Stunde = meinZeitobjekt.getHours();
    var Minute = meinZeitobjekt.getMinutes(); // ist es z.B. 13:01, gibt getMinutes() nur "1" zurück
    if (Minute < 10) {Minute = "0" + Minute};
    var Zeit = Stunde + ":" + Minute;
    document.write("<p>Zeitpunkt Ihrer Anfrage: " + Datum + ", " + Zeit + " Uhr <p>");
</script>
```


© Matthias Gidda, Jens Jacobsen

Text auf der Seite noch eleganter ändern

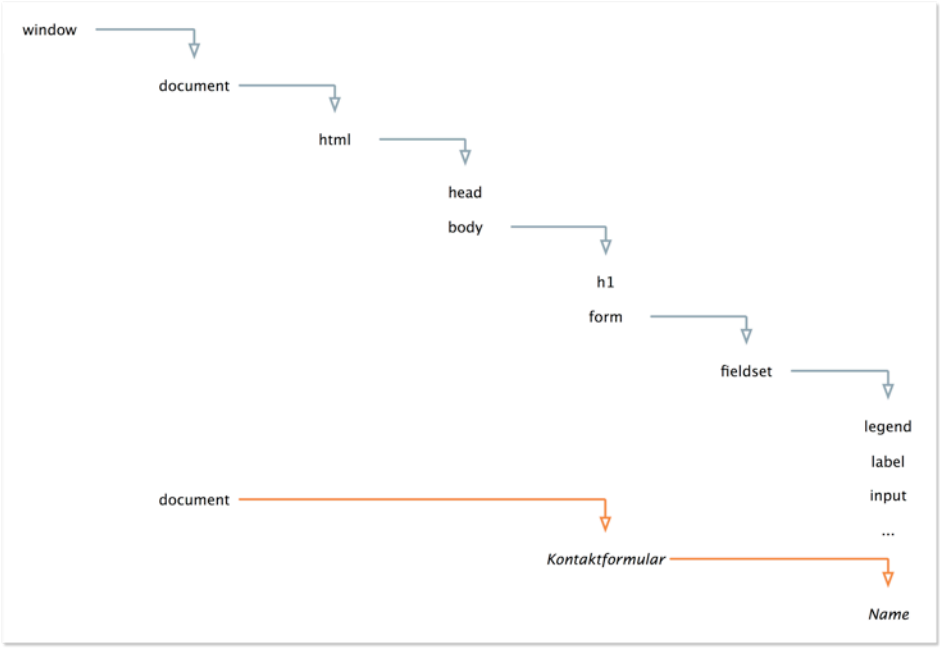
Den Fokus für das Formular haben Sie vorher bei unserem Beispiel mit folgender Zeile gesetzt:

```
window.document.Kontaktformular.Name.focus();
```

`window.` können Sie auch weglassen, wenn Sie das Dokument im aktuellen Fenster ansprechen wollen.

Dabei haben Sie schon ein wichtiges Konzept von JavaScript genutzt, das so genannte document object model, kurz DOM. Das DOM ist sozusagen ein Abbild der aktuellen HTML-Seite. Klingt nicht sehr spannend, ist aber praktisch. Denn Sie können sich im DOM entlanghangeln, um letztlich zu allen Elementen zu kommen, die im Dokument enthalten sind – und diese können Sie dann mit JavaScript verändern.

Das DOM ist wie ein Stammbaum aufgebaut: Das `window` ist der gemeinsame Vorfahre von allen Elementen. Dann kommt `document`, gefolgt von `html`, das sich an dieser Stelle das erste Mal verzweigt in `head` und `body`. Bei unserem Beispielprojekt folgen auf `body` unter anderem die Elemente `h1` und `form`. Letzteres enthält das `fieldset` und darin wiederum liegen `legend`, `label` und mehrere `input`-Elemente.



Die hierarchische Struktur des DOM. Orange sind die Linien unserer Abkürzung, die wir nehmen können, wenn wir Elementen eigene Namen geben.

Sprechen wir ein Element direkt an:

1. Ergänzen Sie nach der Stelle im Code, an der wir nach der Überprüfung der E-Mail-Adresse im Formular den Fokus auf das Feld `email` setzen:

```
document.forms[0].elements[1].style.color = "red";
```

2. Speichern Sie und testen Sie.

Sie sehen jetzt, dass der Text im E-Mail-Feld rot wird, wenn man keine gültige E-Mail-Adresse eingibt und auf „Abschicken“ klickt.

Die Elemente des DOM heißen nodes (=Knoten, so heißen in der Botanik die Ansatzstellen von Ästen am Stamm).

© Matthias Gidda, Jens Jacobsen

Dabei haben wir auf das `document` zugegriffen, dann auf das Element `forms`. Nachdem es mehrere Formulare im Dokument geben kann, gibt JavaScript in dem Fall ein Array zurück (also eine Liste mit allen Formularen, die es enthält). Mit `[0]` sprechen wir den ersten Eintrag im Array an – wir haben auf unserer Seite derzeit nur ein Formular, daher ist es einfach. Dann sprechen wir die `elements` im Formular an, hier gibt es wieder mehrere, daher bekommen wir auch hier ein Array. Wir nehmen diesmal das zweite Element, das wir mit `1` ansprechen, da Arrays per Definition ja immer bei 0 anfangen. Und von diesem Element ändern wir schließlich das Attribut `style`, genauer die `color`, auf `red`.

Es gibt aber auch noch eine andere Möglichkeit, dasselbe zu erreichen:

```
document.forms.Kontaktformular.elements.email.style.
color = "red";
```

In dem Fall können wir auf das Abzählen verzichten. Das ist nicht nur übersichtlicher, es hat auch den Vorteil, dass es noch funktioniert, wenn wir weitere Formulare einfügen, die vor unserem stehen (was die Nummerierung im Array ändern würde). Der obige Ausdruck funktioniert so: Wir gehen vom `document` aus, sprechen darin die Formulare an (`forms`), nehmen hier das mit dem Namen `Kontaktformular`, dort sprechen wir alle enthaltenen Elemente an (`elements`) und wenden uns dort an das mit dem Namen `email`. Bei diesem Ändern wir das Attribut `style` und darin den Wert für die Farbe (`color`) auf Rot (`red`).

Das lässt sich aber noch weiter vereinfachen, da wir eindeutige Namen vergeben haben:

```
document.Kontaktformular.email.style.color = "red";
```

Damit sprechen wir die Elemente direkt mit Namen an und sparen uns so das Entlanghangeln am DOM.

Und noch eine letzte Möglichkeit wollen wir Ihnen zeigen, damit Sie die ganze Bandbreite kennen lernen:

1. Ergänzen Sie im `input`-Element für die E-Mail-Adresse noch ein Attribut:

```
id="email-Feld"
```

2. Ändern Sie die Zeile mit dem Farbwechsel:

```
document.getElementById("email-Feld").style.color = "red";
```

Das ist der kürzeste Weg. Er funktioniert aber nur, wenn Sie eine `id` vergeben haben. Mit `name` geht es deshalb nicht, weil Sie mehreren Elementen in Ihrem Dokument denselben Namen geben dürfen. Haben Sie zum Beispiel zwei Formulare, darf in beiden ein Element mit dem Namen `email` auftauchen. Eine `id` muss aber zwingend einzigartig im Dokument sein.

Eine Verbesserung an anderer Stelle ist, die Formatierung nicht direkt zu ändern, sondern über CSS. Das hat den Vorteil, dass Sie Änderungen an der Farbe später zentral an einer Stelle durchführen können, auch wenn Sie die Formatierung für mehrere Elemente ändern.

1. Ändern Sie die Formatierungs-Zuweisung:

```
document.getElementById("email-Feld").className =
"highlight";
```


© Matthias Gidda, Jens Jacobsen

2. Öffnen Sie die CSS-Datei (wir haben sie *stylesheet8.css* genannt).
3. Ergänzen Sie das folgende Format:

```
.highlight {
   background-color: red;
}
```

Damit erreichen wir nebenbei eine weitere Verbesserung: Es wird jetzt nicht der Text markiert, sondern der gesamte Hintergrund des Textfeldes. Jetzt sieht der Nutzer sofort, dass etwas mit diesem Feld nicht stimmt.

4. Speichern Sie beide Dateien und testen Sie die HTML-Datei im Browser.

Die ganze HTML-Seite finden Sie hier: http://website.benutzerfreun.de/javascript/JS-Formular-5.html

Um das Formular in der Praxis einzusetzen, sollten Sie noch dafür sorgen, dass die rote Markierung wieder verschwindet, wenn der Nutzer eine E-Mail-Adresse eingegeben hat. Ansonsten sieht das schon etwas erschreckend aus. Und außerdem ist das Feld so lange rot hinterlegt, bis der Nutzer den *Abschicken*-Button drückt, auch wenn inzwischen korrekte Werte in dem Feld stehen.

Auch der knallrote Hintergrund passt nicht so recht zur Gestaltung der Seite; das können Sie mit CSS besser!

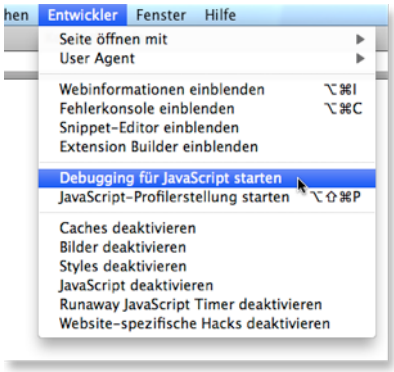
Wenn Sie noch Probleme mit der Umsetzung dieser letzen Korrekturen haben, dann sehen Sie sich die fertige Datei auf dem Server an: `http://website.benutzerfreun.de/javascript/JS-Formular-6.html`

Wie Sie mal wieder gesehen haben, gibt es immer mehrere Möglichkeiten, in JavaScript sein Ziel zu erreichen. Ein Formularelement können Sie zusammenfassend auf diese Weisen ansprechen:

```
document.forms[#].elements[#].eigenschaft

document.forms.name.elements.name.eigenschaft

document.name.name.eigenschaft

document.getElementById("id").eigenschaft
```


© Matthias Gidda, Jens Jacobsen

Auf Fehlersuche mit Firebug/Safari



So faszinierend JavaScript ist, so nervtötend kann es manchmal sein. Jeder kennt das: Wenn man etwas Neues ausprobiert oder Beispielcode anpassen will, will es einfach nicht klappen.

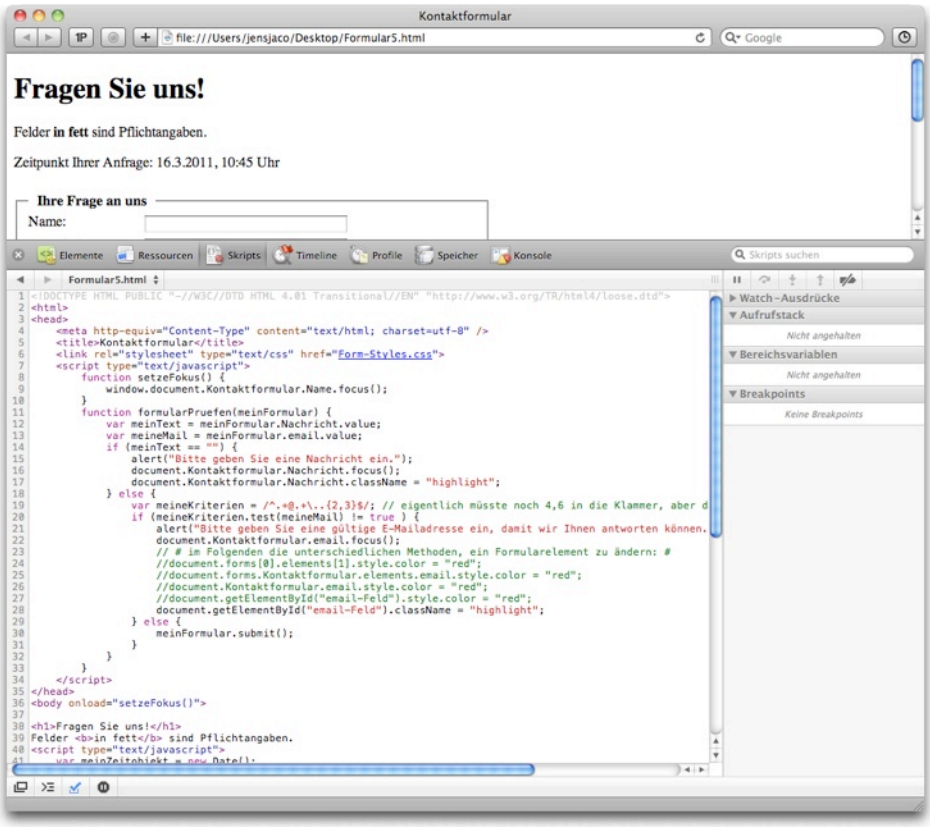
Ein wichtiger Helfer bei der Suche nach Fehlern in eigenen Skripten ist ein so genannter Debugger (wörtlich übersetzt ein „Entfehlerer“ oder „Entwanzer“). Dazu empfehlen sich zwei kostenlose Alternativen:

Entweder Sie nutzen *Safari*, da ist der Debugger schon eingebaut. Oder Sie verwenden *Firebug*, eine kostenlose Erweiterung für Firefox.

Debuggen mit Safari

Gehen Sie in Safari unter Windows auf Bearbeiten und wählen Sie dort Einstellungen. Auf dem Mac finden Sie die Einstellungen unter dem Menüpunkt Safari. Klicken Sie auf Erweitert und setzen Sie das Häkchen bei Menü „Entwickler“ in der Menüleiste anzeigen.

Jetzt ist das gleichnamige Menü zu sehen. Darin finden Sie den Eintrag *Debugging für JavaScript starten*. Wenn Sie das tun, teilt sich das Fenster, und Sie haben Zugriff auf jede Menge Informationen über die gerade geöffnete HTML-Seite. Der wichtigste Bereich für uns ist *Skripts*. Klicken Sie diesen an, sieht es etwa so aus wie in der folgenden Abbildung.



Der Debugger von Safari

Das sieht jetzt noch nicht viel anders aus als in Ihrem HTML-Editor. Zwei große Vorteile eines Debuggers sind aber:

- Sie bekommen aussagekräftige Fehlermeldungen
- Sie können den Wert einzelner Variablen beobachten

© Matthias Gidda, Jens Jacobsen

- Sie können den Code Zeile für Zeile ausführen

Testen wir einmal, was Safari uns über Fehler mitteilt:

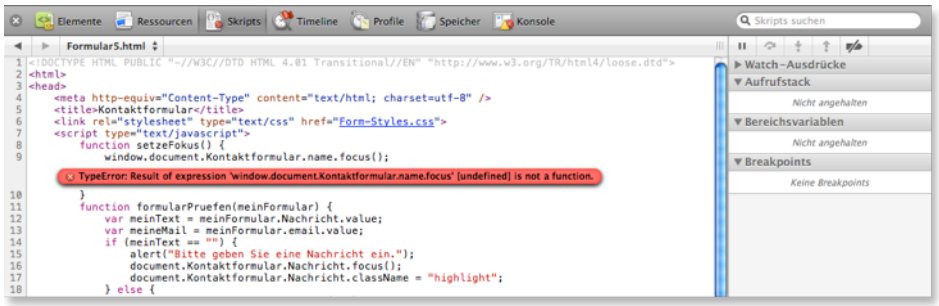
1. Öffnen Sie die letzte Beispielseite mit Ihrem HTML-Editor.
2. Ändern Sie z. B. etwas in der Funktion `onFocus`:

```
window.document.Kontaktformular.name.focus();
```

Haben Sie es gemerkt? Wahrscheinlich nicht, denn das sind die kleinen Fehler, nach denen man ohne Debugger sehr lange sucht: Statt der richtigen Bezeichnung des Eingabefelds `Name` haben wir `name` geschrieben.

3. Speichern Sie die Seite ab, und öffnen Sie sie in Safari.
4. Aktivieren Sie den Debugger im *Entwickler*-Menü.

Sie sehen nun folgende Fehlermeldung:



Der Text ist nicht besonders hilfreich, zugegeben. Aber Sie wissen, dass hier in dieser Zeile etwas nicht stimmt. Und nach etwas Arbeit mit dem Debugger lernen Sie schnell, auch diese Fehlermeldungen richtig zu interpretieren.

Nutzen wir nun den Debugger, um zu verstehen, was JavaScript so denkt:

1. Korrigieren Sie den Fehler im Beispielcode wieder und speichern Sie das Dokument ab.
2. Öffnen Sie es in Safari und aktivieren Sie das Debugging.
3. Klicken Sie auf das kleine Dreieck vor *Watch-Ausdrücke* in der Spalte rechts.
4. Klicken Sie auf *Hinzufügen* und geben Sie `Tag` ein.

Der Watcher zeigt an, welchen Wert diese Variable aktuell hat. Das ist nützlich, wenn Sie im Blick behalten wollen, wie sich der Wert einzelner Variablen während dem Abarbeiten des Skriptes ändert.

Im Watcher können Sie die Werte von Variablen beobachten.

Im letzten Beispiel sehen wir JavaScript bei der Arbeit zu:

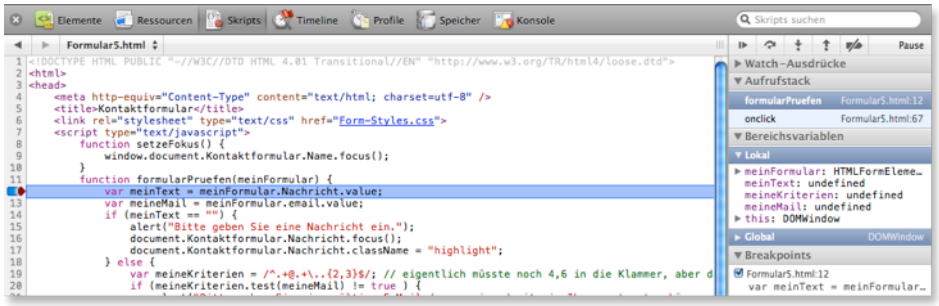
1. Klicken Sie auf die Zeilennummer *12* am linken Bildschirmrand.

Es erscheint dort ein kleiner blauer Pfeil, der anzeigt, dass JavaScript an dieser Stelle anhalten soll. Das nennt man einen Breakpoint.

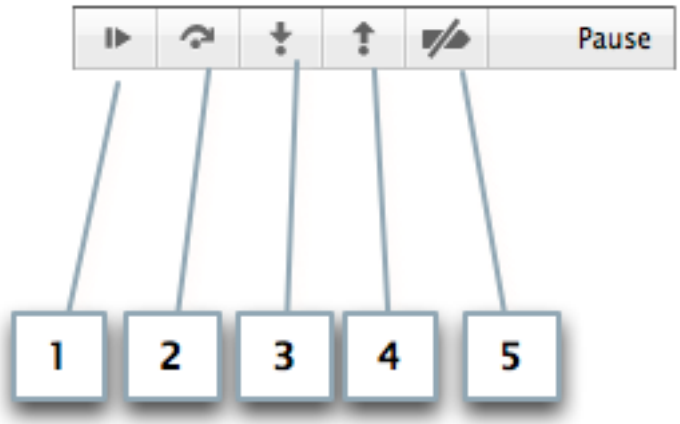
2. Klicken Sie auf den Button *Abschicken* im Formular der HTML-Seite.

Sie sehen, JavaScript hält also tatsächlich an. Der rote Pfeil am linken Rand zeigt Ihnen, wo wir im Skript gerade sind:

© Matthias Gidda, Jens Jacobsen



In der Spalte rechts sehen Sie die Werte aller aktuellen Variablen.



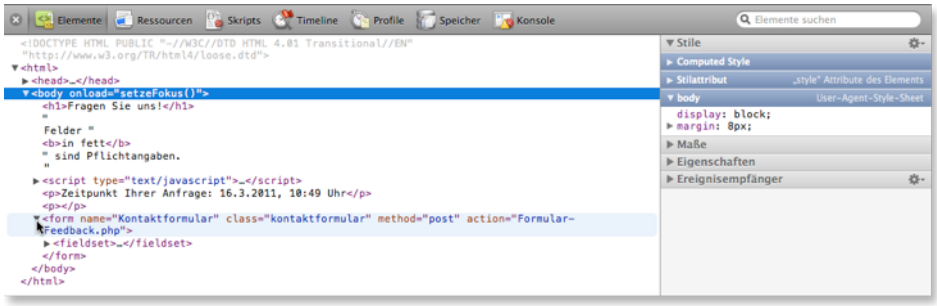
Die Buttons zum Steuern beim schrittweisen Debuggen

Ganz wichtig sind die unbeschrifteten Buttons, die sich in dieser Spalte oben befinden (Details siehe Abbildung oben). Mit diesen sorgen Sie dafür, dass das Skript weiter abgearbeitet wird. Mit dem Button links mit dem Strich und dem Dreieck (1) lassen Sie einfach wieder alles weiterlaufen. Dann kommt der Button, mit dem Sie im Skript vorangehen, aber andere Funktionen überspringen (2). Deshalb ist auch ein Pfeilchen auf dem Button, der über einen Punkt springt. Rufen Sie also in einer Funktion eine andere auf, wird diese bei einem Klick auf diesen Button zwar ausgeführt, Sie sehen das aber nicht im Debugger. Das ist nützlich, wenn Sie sicher sind, dass sich der Fehler in der aktuellen Funktion verstecken muss.

Im Normalfall aber werden Sie den Button daneben benutzen, den mit dem Pfeil, der nach unten auf einen Punkt weist (3). Damit gehen Sie JavaScript Zeile für Zeile durch und sehen, was passiert.

Der Vollständigkeit halber noch eine Erklärung für die beiden nächsten Buttons: Mit dem Pfeil nach oben (4) führen Sie den Rest der aktuellen Funktion aus, ohne es sich anzeigen zu lassen und springen zur nächsten – sofern noch eine im Code kommt. Und der fünfte und letzte Button schließlich (5) schaltet die Breakpoints alle aus, ohne sie zu löschen. Damit können Sie schnell mal überprüfen, wie das Skript laufen würde, ohne anzuhalten. Ein weiterer Klick darauf schaltet alle Breakpoints wieder ein, die Sie gesetzt haben.

© Matthias Gidda, Jens Jacobsen



Im Bereich Elemente können Sie die Teile Ihrer HTML-Seite per Klick auf die kleinen Dreiecke links übersichtlich ein- und ausklappen.

Debuggen mit Firebug

Die Erweiterung *Firebug* für den Browser Firefox kennen Sie aus Kapitel 6. Firebug hat ähnliche Funktionen wie der Debugger von Safari, liefert aber meist aussagekräftigere Fehlermeldungen.

1. Öffnen Sie die letzte Beispielseite mit Ihrem HTML-Editor.
2. Ändern Sie z. B. etwas in der Funktion `onFocus`:

```
window.document.Kontaktformular.name.focus();
```

Statt des korrekten `Name` schreiben wir also `name`.

3. Speichern Sie die Seite ab, und öffnen Sie sie in Firefox.
4. Aktivieren Sie Firebug, indem Sie im Fensterrahmen unten rechts den kleinen Käfer anklicken.
5. Klicken Sie auf den Reiter *Konsole*.

Sie bekommen nun folgende Fehlermeldung:

Sollten Sie das Icon nicht sehen, ist Firebug entweder nicht installiert, oder in seinen Voreinstellungen ist die Anzeige des Icons deaktiviert. Stellen Sie auch sicher, dass die Statusleiste angezeigt wird (im Menü Ansicht).

Sie sehen: Diese Fehlermeldung ist deutlich aussagekräftiger als die von Safari. Sie erfahren von Firebug, dass das Problem bei der Funktion liegt. Trotzdem kommt man als Neuling wahrscheinlich nicht sofort darauf, was genau am eigenen Skript nicht stimmt. Aber wenn Sie selbst mal eine Weile nach so einem Tippfehler gesucht haben, dann werden Sie es für die nächsten Male wissen.

Mit einem Klick auf die blaue Angabe der Fehlerquelle rechts springen Sie übrigens direkt in den Quellcode (entspricht einem Klick auf den Reiter *Skript*).

Ansonsten bietet Firebug alle Funktionen von Safari. Im Bereich *Skript* können Sie wie in Safari Breakpoints setzen, nur heißen sie hier Haltepunkte. Variablen überwachen können Sie mit einem Klick auf *Neuer Überwachungsausdruck…* in der rechten Spalte des Skript-Fensters.

Sehr nützlich ist der Bereich *Konsole*, weil man hier einzelne JavaScript-Schnipsel direkt testen kann. (Eine Konsole gibt es auch bei Safari.)

1. Klicken Sie dazu auf das kleine Dreieck im grauen Kreis, das direkt über dem Firebug-Käfer unten rechts auf dem Fensterrahmen sitzt.

© Matthias Gidda, Jens Jacobsen

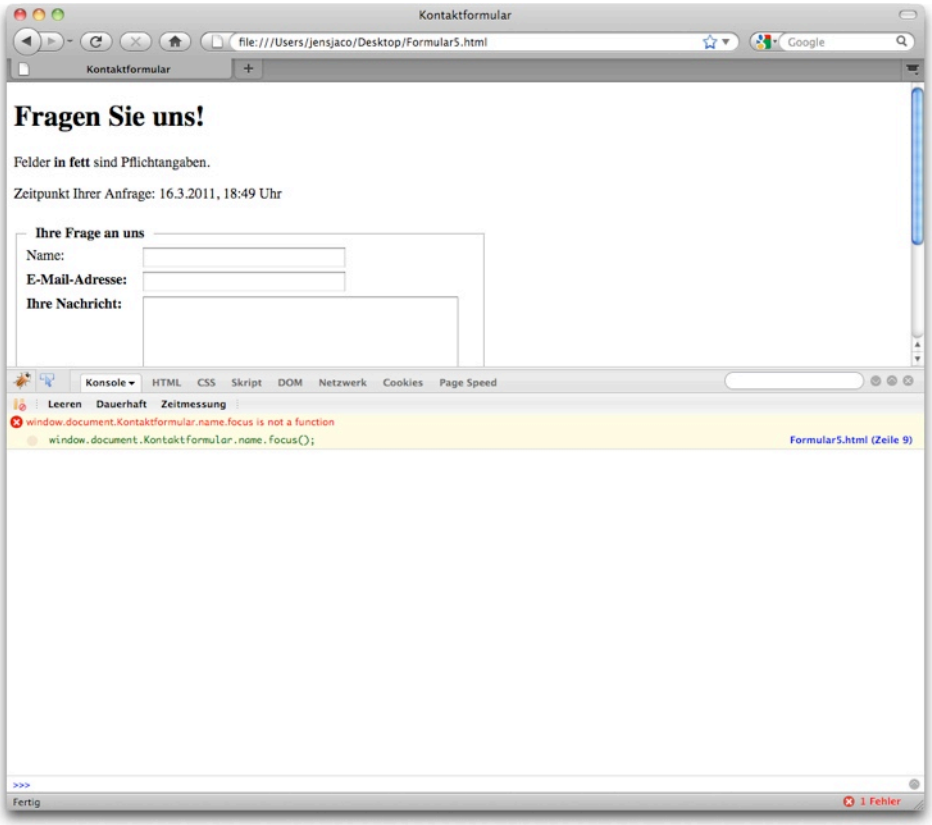
Dadurch erscheint eine zweite Spalte, in der Sie direkt Text eingeben können.

2. Tippen Sie dort:

```
console.log("Hallo, Welt!");
```

3. Klicken Sie auf *Ausführen* unten links in der rechten Spalte.

Sie sehen, mit diesem Befehl können Sie einfach kleine JavaScript-Stücke testen:



Mit diesem Werkzeug bewaffnet, finden Sie die Bugs in Ihren Skripten deutlich leichter, als wenn Sie immer wieder und wieder Ihren Code durchlesen.

© Matthias Gidda, Jens Jacobsen

HTML5

HTML5 ist die nächste Version von HTML. Leider unterstützen nur die neuesten Browser HTML5, weshalb man heute noch nicht auf reine HTML5-Lösungen setzen sollte. Aber wir können uns schon darauf freuen, es in Zukunft leichter zu haben. Einige Eigenschaften von HTML5 machen JavaScript bei Formularen fast überflüssig: Formulare haben darin zum Beispiel das Attribut `autofocus`. Es sorgt dafür, dass der Cursor automatisch im entsprechenden Element landet.

Das sieht dann so aus:

```
<input type="text" name="benutzername" autofocus>
```

Wir halten Sie über die aktuelle Entwicklung auf dem Laufenden unter http://website.benutzerfreun.de/html5/.

Bibliotheken wie jQuery verwenden

Wie Sie gesehen haben, ist JavaScript zwar sehr mächtig, aber manchmal etwas umständlich. Gerade wenn man immer wieder Standard-Aktionen machen muss, ist es mühsam, jedes Mal den Code von Neuem zu schreiben.

Download und Dokumentation von jQuery finden Sie hier: http://jquery.com

Sie haben ja gesehen, dass man JavaScript-Funktionen auch in eigene Dateien auslagern und mit HTML-Dokumenten verlinken kann. So können Sie Funktionen in allen Seiten Ihrer Site nutzen und sie auch in anderen Projekten problemlos wiederverwenden.

Sie können aber auch auf ganze Funktionssammlungen zurückgreifen, die andere geschrieben haben. Davon gibt es eine ganze Menge, die Sie kostenlos verwenden können.

Wohl die bekannteste JavaScript-Bibliothek ist *jQuery*. Sie zu nutzen ist einfach:

1. Sie laden die Datei jquery.js herunter.
2. Sie übertragen sie auf Ihren Webserver.
3. Sie binden Sie im HTML-Dokument ein mit:

```
<script src="jquery.js"></script>
```

4. Jetzt können Sie jQuery-Befehle nutzen. Z. B.:

```
$("p").click(function() { anweisungen... });
```

Damit bekommen alle `p`-Elemente des Dokuments einen `onClick`-Handler, der den Code ausführt, der in der Klammer nach dem `click` steht – das alles erreichen Sie mit nur einer Zeile Code.

Es dauert ein bisschen, bis man sich in jQuery eingearbeitet hat, aber wenn Sie mehr mit JavaScript machen wollen, dann sparen Sie sich mittelfristig viel Arbeit.

© Matthias Gidda, Jens Jacobsen