



# HTML5

› Leitfaden für Webentwickler



Egal ob Sie einen Flug buchen, Ihre Bankgeschäfte online abwickeln oder einen Suchbegriff bei Google eingeben – ohne Formularfelder wären diese Dienste nicht nutzbar. Seit der Version 2.0 von HTML aus dem Jahr 1995 sind die meisten Elemente für interaktive Formulare unverändert, was einerseits für ein sehr vorausschauendes Design von Tim Berners Lee spricht; andererseits hat sich dadurch aber auch ein großer Nachholbedarf entwickelt. Die HTML5-Spezifikation widmet dem Thema Formulare einen großen Abschnitt und wird die Arbeit von Webentwicklern drastisch erleichtern.

Auch wenn zu dem Zeitpunkt, als dieses Buch geschrieben wurde, die Unterstützung der Browser noch nicht besonders weit gediehen war (einzig Opera

und die Entwicklerversion von Google Chrome sind hier zu erwähnen), ist es durch die rückwärtskompatible Syntax möglich, die neuen Formular-Elemente schon heute unbedenklich einzusetzen.

## 3.1 Neue Input-Typen

Die HTML5-Spezifikation wertet das `input`-Element deutlich auf, indem es für das `type`-Attribut mehrere neue Werte vorsieht. Die neuen Typen wie `date`, `color` oder `range` machen es einerseits für Browserhersteller möglich, bedienerfreundliche Eingabeelemente zur Verfügung zu stellen, andererseits kann der Browser sicherstellen, dass die Eingaben vom gewünschten Typ sind. Erkennt ein Browser den `type` des `input`-Elements nicht, so fällt er auf `type=text` zurück und zeigt ein Textfeld an, was in jedem Fall hilfreich ist. Dieses Verhalten zeigen auch ältere Browser, wodurch dem Einsatz der neuen Typen nichts im Wege steht.

Den größten Nutzen werden wohl die Typen für Datum und Uhrzeit mit sich bringen. Aktuell kursieren unzählige verschiedene Versionen von mehr oder weniger gelungenen JavaScript-Kalendern im Internet. Egal ob Sie einen Flug buchen, ein Hotel reservieren oder sich bei einer Tagung anmelden, die komfortable Eingabe eines Datums ist ein Problem, das bisher immer Handarbeit verlangte. Natürlich bieten JavaScript-Bibliotheken wie *jQuery* fertige Kalender an, aber eigentlich sollte diese Funktion vom Browser direkt unterstützt werden.

Im Sommer 2010 gab es nur einen Desktop-Browser, der ein grafisches Bedienungselement für die Datumseingabe mitlieferte, nämlich Opera. Abbildung 3.1 zeigt den aufgeklappten Kalender, der beim Anklicken eines `input`-Elements vom Typ `date` erscheint. Aber der Reihe nach – zuerst verschaffen wir Ihnen in Tabelle 3.1 einen Überblick über die neuen Typen und zeigen Ihnen dann in Abbildung 3.1 deren Umsetzung im Opera-Browser.

Typ	Beschreibung	Beispiel
<code>tel</code>	Text ohne Zeilenumbrüche	+49 6473 3993443
<code>search</code>	Text ohne Zeilenumbrüche	suchbegriff
<code>url</code>	eine absolute URL	<i>http://www.example.com</i>

Typ	Beschreibung	Beispiel
email	eine gültige E-Mail-Adresse	<i>user@host.com</i>
datetime	Datum und Uhrzeit (immer in der UTC- Zeitzone)	2010-08-11T11:58Z
date	Datumsangabe ohne Zeitzone	2010-08-11
month	Monatsangabe ohne Zeitzone	2010-08
week	Jahr und Woche im Jahr ohne Zeitzone	2010-W32
time	Uhrzeit ohne Zeitzone	11:58
datetime-local	Datum und Uhrzeit ohne Angabe einer Zeitzone	2010-08-11T11:58:22.5
number	Zahl	9999 oder 99.2
range	Numerischer Wert eines Wertebereichs	33 oder 2.99792458E8
color	Hexadezimale Darstellung von RGB-Werten im sRGB-Farbraum	#eeeeee

**Tabelle 3.1:** Neue Input-Typen in HTML5

### 3.1.1 Die Input-Typen »tel« und »search«

tel und search unterscheiden sich nicht wesentlich von normalen Textfeldern. In beiden sind Zeichenketten ohne Zeilenumbrüche zulässig. Auch Telefonnummern sind nicht auf Zahlen beschränkt, da hier oft Klammern oder das Pluszeichen verwendet wird. Bei tel könnte der Browser Vorschläge aus dem lokalen Adressbuch anbieten, eine Situation, die vor allem auf Mobiltelefonen sehr nützlich sein kann. Der search-Typ wurde eingeführt, damit der Browser die Sucheingabe in einem konsistenten Layout zur jeweiligen Plattform gestalten kann. Zum Beispiel sind Benutzer des Mac OS X-Betriebssystems an abgerundete Ecken bei Suchfeldern gewohnt.

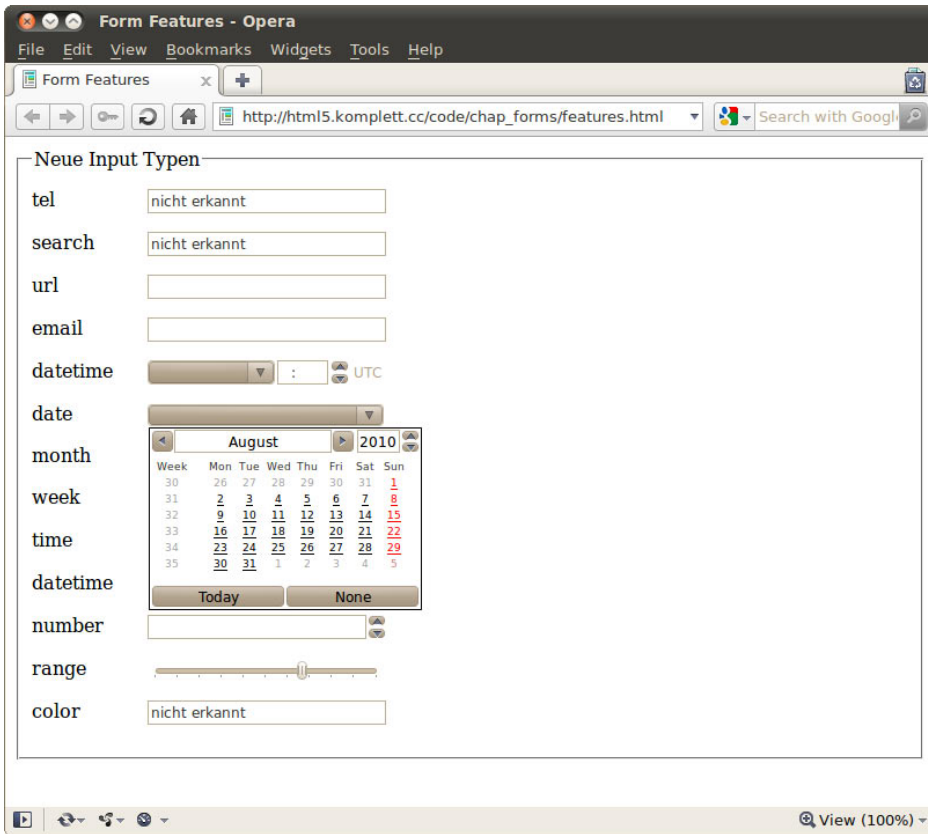


Abbildung 3.1: Opera ist bei der Implementierung von neuen Formular-Input-Typen bereits sehr weit.

### 3.1.2 Die Input-Typen »url« und »email«

Bei `url` und `email` ist außer möglichen Vorschlägen auch eine Prüfung der Syntax möglich. Da es sowohl für E-Mail-Adressen als auch für Internet-Adressen in Form von URLs konkrete Vorschriften gibt, kann der Browser bereits während der Eingabe eine Rückmeldung über mögliche Fehler geben (mehr dazu in Abschnitt 3.4, Clientseitige Formular-Überprüfung).

### 3.1.3 Datum und Uhrzeit mit »datetime«, »date«, »month«, »week«, »time« und »datetime-local«

Die Datums- und Uhrzeitformate bedürfen einer genaueren Betrachtung. `datetime` enthält Datumsangabe und Uhrzeit, wobei als Zeitzone immer UTC

verwendet wird. Laut Spezifikation kann der Browser den Anwender auch eine andere Zeitzone auswählen lassen, der Wert des `input`-Elements muss aber in UTC umgerechnet sein. Die Regeln für Zeitangaben beim `datetime`-Attribut des `time`-Elements, die wir in Abschnitt 2.7.2, Das Element »time«, erklären, treffen auch hier zu – mit der einzigen Ausnahme, dass die Zeichenkette immer mit `Z`, dem Bezeichner für UTC, abgeschlossen werden muss.

Bei `date` und `month` fällt die Angabe der Uhrzeit und der Zeitzone weg. Für `date` wird in der Spezifikation außerdem erwähnt, dass es sich um eine gültige Tagesangabe innerhalb des ausgewählten Monats handeln muss, wobei Schaltjahre mit einzubeziehen sind. Jahr, Monat und Tag sind mit einem Minuszeichen zu trennen, wobei die Jahresangabe mindestens vier Stellen enthalten und größer als 0 sein muss. Damit sind, anders als in dem etwas ausführlicheren ISO-Standard 8601, keine Zeitpunkte vor Christus darstellbar.

Der Typ `week` wird als Woche im Jahr dargestellt, und ihm wird zwingend das Jahr vorangestellt. Als Trennzeichen zwischen Jahr und Woche dient abermals das Minuszeichen. Damit die Angabe nicht mit der von `month` verwechselt werden kann, muss der Woche das Zeichen `W` vorangestellt werden.

`datetime-local` funktioniert analog zu dem oben beschriebenen `datetime`, mit dem einzigen Unterschied, dass die Angabe der Zeitzone entfällt.

Opera verwendet für die Auswahl aller Datumsangaben ein Kalenderfenster; die Angaben zur Uhrzeit können manuell eingegeben oder über Pfeiltasten am Rand verändert werden (vergleiche Abbildung 3.1).

### 3.1.4 Die Input-Typen »number« und »range«

Die Typen `number` und `range` verlangen, dass die Eingabe in einen numerischen Wert umgewandelt werden kann, wobei auch die Notation für Gleitkommazahlen (zum Beispiel `2.99792458E8`) gültig ist. Für den `range`-Typ enthält die Spezifikation die Anmerkung, dass der genaue Wert nicht entscheidend ist. Es handelt sich um eine ungefähre Angabe, die vom Anwender gut mit einem Schieberegler eingestellt werden kann. Sowohl Webkit-basierte Browser wie Safari und Google Chrome als auch Opera verwenden zur Darstellung dieses Typs einen Schieberegler (vergleiche Abbildung 3.1 und Abbildung 3.2).

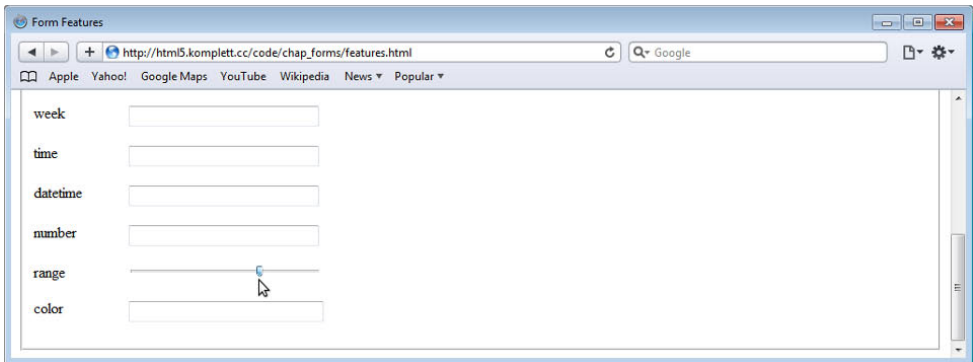


Abbildung 3.2: Safari mit dem Input-Typ »range«

### 3.1.5 Der Input-Typ »color«

Leider gänzlich ohne Implementierung ist der neue Typ `color`. Ähnlich wie bei dem Typ `date` gibt es auch hier schon etliche Versuche in JavaScript; im Vergleich zum Datum ist die Farbauswahl aber wesentlich seltener notwendig. Zukünftige Implementierungen im Browser werden aber wahrscheinlich einen Farbwähler vorsehen, wie man ihn von Bildbearbeitungsprogrammen her kennt. Der Wert für das `input`-Element muss die 8-Bit-Rot-, -Grün- und -Blau-Werte in hexadezimaler Notation mit führendem `#`-Zeichen enthalten. Die Farbe Blau wird in dieser Notation zum Beispiel als `#0000ff` geschrieben.

### 3.1.6 Die neuen Input-Typen im Einsatz

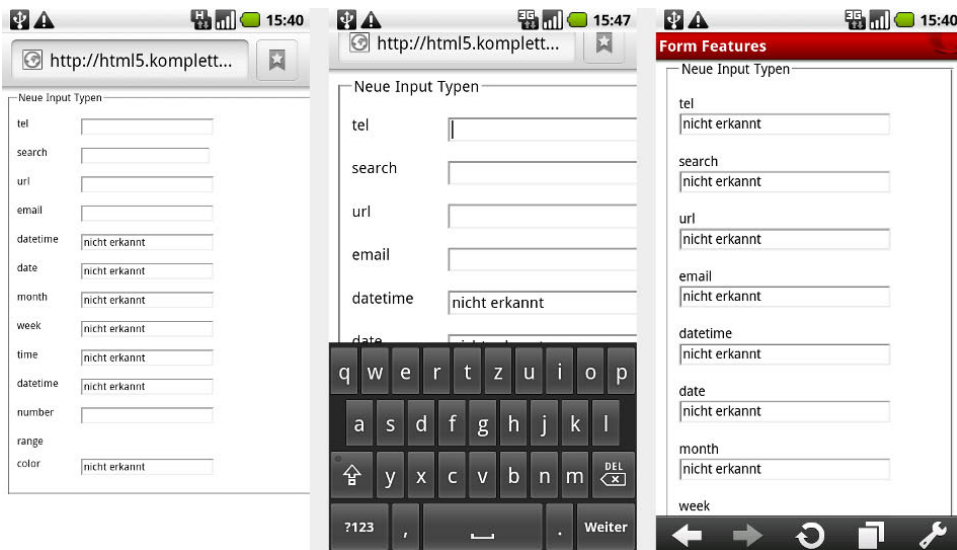
Genug der Theorie. In einem ersten Beispiel werden alle neuen Elemente untereinander dargestellt. Da das allein noch keine besondere Herausforderung darstellt, soll jedes Element noch auf seine Funktion geprüft werden. Der Trick dabei ist, dass der Browser den Typ eines unbekanntes Elements auf `text` setzt, und diese Eigenschaft können wir in JavaScript abfragen:

```
<script>
  window.onload = function() {
    inputs = document.getElementsByTagName("input");
    for (var i=0; i<inputs.length; i++) {
      if (inputs[i].type == "text") {
        inputs[i].value = "nicht erkannt";
      }
    }
  }
</script>
```

Sobald die Webseite vollständig geladen ist, läuft eine Schleife über alle `input`-Elemente, in der deren `type`-Attribute analysiert werden. Sofern das `type`-Attribut dem Standard-Typ `text` entspricht, wird dessen Wert auf *nicht erkannt* gesetzt. Der HTML-Code für die neuen `input`-Elemente sieht folgendermaßen aus:

```
<fieldset>
  <legend>Neue Input-Typen</legend>
  <p><label for=tel>tel</label>
  <input type=tel id=tel name=tel>
  <p><label for=search>search</label>
  <input type=search id=search name=search>
  <p><label for=url>url</label>
  <input type=url id=url name=url>
  <p><label for=email>email</label>
  ...
```

Wie das Ergebnis dieses Tests auf einem Android-Mobiltelefon ausfällt, zeigt Abbildung 3.3. Der Webkit-basierte Browser des Systems (links) gibt zwar vor, die Typen `tel`, `search`, `url` und `email` zu kennen, leistet aber bei der Eingabe der Telefonnummer über die Tastatur (Mitte) keine besondere Hilfe. Opera Mini in Version 5.1 (rechts) gibt direkt zu, dass es keinen der neuen Typen unterstützt.

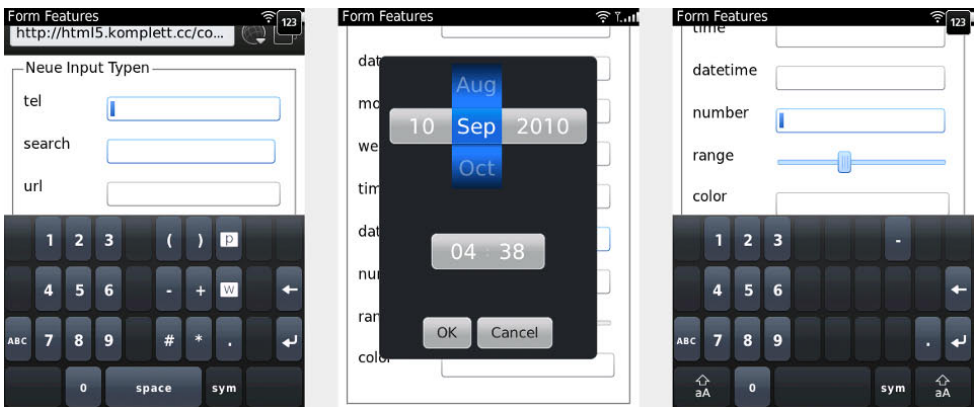


**Abbildung 3.3:** Unterstützung neuer Formular-Input-Typen auf einem Android 2.1-Telefon mit den Browsern Webkit (links, Mitte) und Opera (rechts)



Das ist eine enttäuschende Bilanz für die sonst so modernen mobilen Browser. Auf dem iPhone sieht die Sache etwas besser aus: So passt das Smartphone zumindest die Software-Tastatur so an, dass bei der Eingabe von Telefonnummern ein Zahlenfeld erscheint, und bei dem `input`-Typ `email` wird das `@`-Zeichen auf der Tastatur hinzugefügt.

Noch etwas besser funktioniert das mit BlackBerry, dem Betriebssystem des kanadischen Smartphone-Herstellers *Research in Motion (RIM)*, dessen Geräte vor allem in Nordamerika weit verbreitet sind. Wie Abbildung 3.4 zeigt, werden sowohl `tel` als auch `number` und Datumstypen unterstützt, wobei vor allem Letztere grafisch sehr ansprechend aufbereitet sind. Unter der Haube arbeitet *Webkit*, wobei die Software um diese Funktionen erweitert wurde.



**Abbildung 3.4:** Die neuen `input`-Typen auf einem BlackBerry-Smartphone (BlackBerry 9800-Simulator)

## 3.2 Nützliche Attribute für Formulare

Neben neuen Elementen und vielen neuen Typen für das `input`-Element bietet HTML5 auch einige neue Attribute für Formular-Elemente.

### 3.2.1 Fokussieren mit »autofocus«

Google überraschte viele Anwender vor Jahren mit einem einfachen Trick, der die Suchseite deutlich komfortabler machte: Beim Laden der Seite positionierte sich der Cursor automatisch im Suchfeld. Dadurch konnte man unmittelbar den Suchbegriff eingeben, ohne erst mit der Maus das Eingabefeld aktivieren

zu müssen. Was bisher mit einem kurzen JavaScript-Schnipsel erledigt wurde, kann in HTML5 mit dem `autofocus`-Attribut erreicht werden.

```
<input type=search name=query autofocus>
```

Wie alle Attribute vom Typ *boolean* kann das Attribut auch als `autofocus="autofocus"` geschrieben werden (vergleiche dazu Abschnitt 2.7.2, Das Element »time«). Laut Spezifikation darf nur ein Element einer Webseite das `autofocus`-Attribut enthalten.

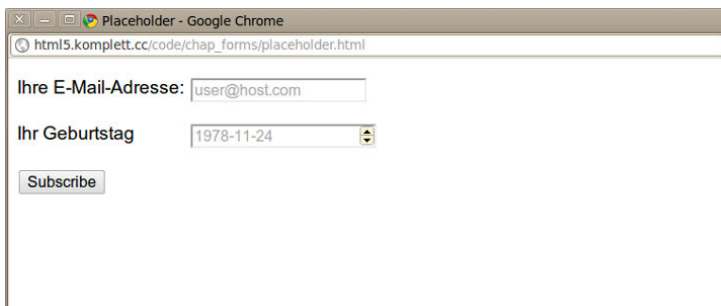
Für ältere Browser stellt `autofocus` kein Hindernis dar, weil sie das unbekannte Attribut einfach ignorieren. Den Gewinn an Benutzerfreundlichkeit haben freilich nur neue Browser.

### 3.2.2 Platzhalter-Text mit »placeholder«

Eine weitere Verbesserung der Benutzbarkeit von HTML-Formularen erreicht man durch das neue `placeholder`-Attribut.

```
<p><label for=email>Ihre E-Mail-Adresse:</label>
<input type=email name=email id=email
  placeholder="user@host.com">
<p><label for=birthday>Ihr Geburtstag</label>
<input type=date name=birthday id=birthday
  placeholder="1978-11-24">
```

Der Wert von `placeholder` kann dem Benutzer einen kurzen Hinweis darauf geben, wie das Feld auszufüllen ist, und sollte nicht als Ersatz für das `label`-Element verwendet werden. Das bietet sich vor allem bei solchen Feldern an, wo ein bestimmtes Eingabeformat erwartet wird. Der Browser zeigt den Hinweistext innerhalb eines nicht aktiven Eingabefelds an. Sobald das Feld aktiviert wird und den Fokus erhält, wird der Text nicht mehr angezeigt (vergleiche Abbildung 3.5).



**Abbildung 3.5:** Das »placeholder«-Attribut in Google Chrome

### 3.2.3 Verpflichtende Felder mit »required«

`required` ist ein *boolean*-Attribut, das mit einem Wort schon alles über seine Funktion verrät: Ein Formular-Element, dem dieses Attribut zugewiesen ist, muss ausgefüllt werden. Wenn ein `required`-Feld beim Abschicken des Formulars leer ist, so erfüllt es nicht die erforderlichen Vorgaben, und der Browser muss darauf entsprechend reagieren. Mehr dazu finden Sie in Abschnitt 3.4, Clientseitige Formular-Überprüfung.

### 3.2.4 Noch mehr neue Attribute für das »input«-Element

Das `input`-Element wurde nicht nur durch neue Typen (Abschnitt 3.1, Neue Input-Typen) aufgewertet, sondern auch durch neue Attribute, die die Handhabung von Formularen erleichtern.

Attribut	Typ	Beschreibung
<code>list</code>	String	Verweis auf die ID eines <code>datalist</code> -Elements mit Vorschlägen (vergleiche Abschnitt 3.3.3, Auswahllisten mit » <code>datalist</code> «)
<code>min</code>	Numerisch/Datum	Minimalwert für numerische Felder und Datumsfelder
<code>max</code>	Numerisch/Datum	Maximalwert für numerische Felder und Datumsfelder
<code>step</code>	Numerisch	Schrittweite für numerische Felder und Datumsfelder
<code>multiple</code>	Boolean	Mehrfachauswahl möglich
<code>autocomplete</code>	Enumerated (on/off/default)	Vorausfüllen von Formularfeldern mit gespeicherten Daten
<code>pattern</code>	String	Regulärer Ausdruck zum Überprüfen des Werts

**Tabelle 3.2:** Neue Attribute für das »input«-Element

Dem `list`-Attribut werden wir noch in Abschnitt 3.3.3, Auswahllisten mit »`datalist`«, begegnen. Es verweist auf das `datalist`-Element, das mögliche Einträge als Vorschläge bereitstellt.

`min`, `max` und `step` eignen sich nicht nur für numerische Felder; auch bei Datums- und Zeitangaben können diese Attribute verwendet werden.

```
<p><label for=minMax>Zahlen zwischen 0 und 1:</label>
<input type=number name=minMax id=minMax
  min=0 max=1 step=0.1>
<p><label for=minMaxDate>Datum mit Wochenschritten:</label>
<input type=date name=minMaxDate id=minMaxDate
  min=2010-08-01 max=2010-11-11 step=7>
<p><label for=minMaxTime>Zeit mit Stundenschritten:</label>
<input type=time name=minMaxTime id=minMaxTime
  min=14:30 max=19:30 step=3600>
```

In Browsern, die den `input`-Typ `number` unterstützen, wird das erste `input`-Element (`id=minMax`) jeweils um den Wert von 0.1 erhöht. Das funktioniert durch den Klick auf die Pfeiltasten am Ende des Textfeldes oder durch das Drücken der Pfeiltasten auf der Tastatur. Das Element mit der ID `minMaxDate` springt jeweils um sieben Tage weiter. Opera zeigt dabei in dem Kalender nur jene Tage als aktiv an, die dem Wochenzyklus entsprechen. Google Chrome bietet zum Einstellen dieses Elements die gleiche Navigation wie beim `input`-Typ `number`: zwei Pfeiltasten, die das Datum um sieben Tage vor oder zurück stellen. Bei dem dritten `input`-Element in diesem Beispiel wird die Schrittweite mit 3600 angegeben, was dazu führt, dass die Zeitangabe jeweils um eine Stunde vor oder zurückgestellt wird. Obwohl in der Spezifikation erwähnt ist, dass die Eingabelemente für Zeitangaben normalerweise mit einer Genauigkeit von Minuten arbeiten, interpretieren sowohl Opera als auch Google Chrome diese Angabe als Sekunden.

Die Mehrfachauswahl ist uns allen vom Kopieren von Dateien her bekannt; im Browser gibt es diese Möglichkeit jetzt auch. Wollte man bisher mehrere Dateien auf einer Webseite laden, so musste man für jede Datei ein `input`-Feld vorsehen. Das `multiple`-Attribut ermöglicht es, im Dateialog mehrere Dateien zu markieren. Für das `select`-Element war die `multiple`-Option schon immer vorgesehen, neu ist die Verwendung für Eingabefelder vom Typ `email`. Im Sommer 2010 konnte aber keiner der gängigen Desktop-Browser diese Funktion für `email`-Typen umsetzen.


Moderne Browser verfügen über eine Funktion, durch die Formular-Eingaben gespeichert werden, damit sie bei einem neuerlichen Zugriff auf das Formular als Hilfe beim Ausfüllen dienen. Was meist sehr praktisch ist, kann bei sicherheitskritischen Eingabefeldern auch unerwünscht sein (die Spezifikation erwähnt hier als Beispiel die Abschusscodes von Nuklearwaffen). Das `autocom-`

plete-Attribut wurde eingeführt, damit Webentwickler dieses Verhalten steuern können. Wird ein Element mit dem Attribut `autocomplete="off"` versehen, so bedeutet das, dass die eingegebene Information vertraulich ist und nicht im Browser gespeichert werden soll. Enthält ein Formular-Element keinen Hinweis, ob `autocomplete` ein- oder ausgeschaltet sein soll, so ist der Standardwert, dass Vorschläge angezeigt werden sollen. Das `autocomplete`-Attribut kann auch auf das ganze Formular angewendet werden, indem man es dem `form`-Element zuweist.

Um eine sehr flexible Überprüfung der Eingabe zu ermöglichen, wurde das `pattern`-Attribut eingeführt. Durch die Angabe eines *regulären Ausdrucks* wird das Formularfeld auf eine Übereinstimmung geprüft. Reguläre Ausdrücke stellen eine sehr mächtige, aber leider auch nicht ganz einfache Methode zur Behandlung von Strings dar. Stellen Sie sich vor, Sie suchen eine Zeichenkette, die mit einem Großbuchstaben beginnt, auf den eine beliebige Anzahl von Kleinbuchstaben oder Zahlen folgt, und die auf `.txt` endet. Mit einem *regex* (eine Kurzform für *Regular Expression*, d. h. *regulärer Ausdruck*) ist das kein Problem:

```
[A-Z]{1}[a-z,0-9]+\ .txt
```

### HINWEIS



Eine Einführung in reguläre Ausdrücke würde den Rahmen dieses Kapitels bei Weitem sprengen, also setzen wir für den folgenden Abschnitt Grundkenntnisse in regulären Ausdrücken voraus. Wenn Sie eine kurze Online-Einführung in *reguläre Ausdrücke* suchen, sind Sie natürlich bei Wikipedia gut beraten:

[http://de.wikipedia.org/wiki/Regulärer\\_Ausdruck](http://de.wikipedia.org/wiki/Regulärer_Ausdruck)

Die Webseite <http://www.regexe.de/> bietet eine interessante Anleitung und die Möglichkeit, reguläre Ausdrücke gleich online auszuprobieren.

Beim Einsatz von regulären Ausdrücken im `pattern`-Attribut ist zu beachten, dass das Suchmuster immer auf den gesamten Inhalt des Feldes zutreffen muss. Außerdem wird in der Spezifikation vorgeschlagen, dass das `title`-Attribut dazu verwendet wird, dem Anwender einen Hinweis zu geben, wie das Format der Eingabe ist. Opera und Google Chrome zeigen diese Informationen dann in Form eines Tool-Tipps an, sobald sich der Mauszeiger über dem Feld befindet. Nach so viel Theorie folgt nun endlich ein kurzes Beispiel:

```
<p><label for=pattern>Ihr Nickname:</label>  
<input type=text pattern="[a-z]{3,32}"  
  placeholder="johnsmith" name=pattern id=pattern  
  title="Bitte nur Kleinbuchstaben, min. 3, max. 32!">
```

Die Vorgabe für das `pattern` lautet, dass die Zeichenkette nur Zeichen zwischen `a` und `z` (also Kleinbuchstaben) enthalten darf (`[a-z]`) und davon mindestens 3 und höchstens 32. Umlaute und andere Sonderzeichen sind damit nicht erlaubt, was für einen Benutzernamen, wie in oben stehendem Beispiel, auch ganz gut sein kann. Wollte man zumindest die deutschen Umlaute und das scharfe `ß` mit einbeziehen, müsste man die Gruppe um diese erweitern: `[a-zäöüß]`. Was passiert, wenn die Überprüfung fehlschlägt, wird in Abschnitt 3.4, Clientseitige Formular-Überprüfung, weiter ausgeführt.

## 3.3 Neue Elemente

### 3.3.1 Anzeigen von Messgrößen mit »meter«

Mithilfe des `meter`-Elements wird der Anteil an einer gewissen Größe grafisch dargestellt. Denken Sie zum Beispiel an die Tankanzeige in Ihrem Auto: Die Nadel zeigt den aktuellen Füllstand im Tank irgendwo zwischen 0 und 100 Prozent an. Bisher wurden solche grafischen Darstellungen in HTML unter anderem mit verschachtelten `div`-Elementen codiert, eine relativ unelegante Lösung, die wohl etwas am Sinn des `div`-Elements vorbeigeht. Außerdem lässt sich eine Statusanzeige auch grafisch, als Bild darstellen. Dabei können freie Webservices herangezogen werden, wie zum Beispiel die *Google Chart API*. Alle diese Möglichkeiten werden Sie im folgenden Beispiel sehen.

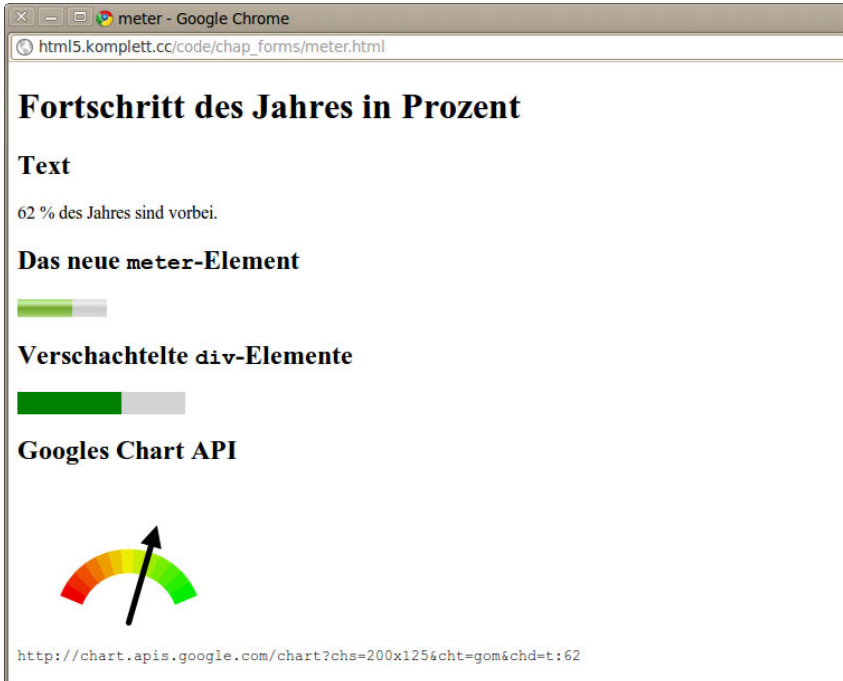
Die Verwendung des `meter`-Elements ist sehr einfach: Über das `value`-Attribut wird der gewünschte Wert eingestellt; alle anderen Attribute sind optional. Wird kein `min`- und `max`-Wert eingestellt, so verwendet der Browser 0 beziehungsweise 1 für diese Attribute. Folgendes `meter`-Element zeigt also ein halb volles Element an:

```
<meter value=0.5></meter>
```

Außer `value`, `min` und `max` gibt es noch die Attribute `low`, `high` und `optimum`, wobei der Browser diese Werte in der Darstellung mit einbeziehen kann. So zeigt zum Beispiel Google Chrome (im Sommer 2010 der einzige Browser, der das `meter`-Element darstellen konnte) den ansonsten grünen Balken in Gelb an, wenn der `optimum`-Wert überschritten wird.

Im folgenden Beispiel wird der Anteil der vergangenen Tage im aktuellen Jahr grafisch dargestellt. Die Webseite soll die Ausgabe auf vier verschiedene Arten visualisieren: als Text mit einer Angabe in Prozent, mithilfe des neuen `meter`-

Elements, durch verschachtelte `div`-Elemente und als Grafik, die durch den Webservice von Googles Chart API erzeugt wird. Das Resultat sehen Sie in Abbildung 3.6.



**Abbildung 3.6:** Das »meter«-Element und ähnliche Möglichkeiten zur Darstellung eines Zustandes

Der HTML-Code für das Beispiel enthält die noch leeren Elemente, die mithilfe von JavaScript befüllt werden:

```
<h2>Text</h2>
<p><output id=op></output><span id=opText></span>
  % des Jahres sind vorbei.</p>
<h2>Das neue <span class=tt>meter</span>-Element</h2>
<meter value=0 id=m></meter>
<h2>Verschachtelte <span class=tt>div</span>-Elemente</h2>
<div id=outer style="background:lightgray;width:150px;" >
<div id=innerDIV>&nbsp;</div></div>
<h2>Googles Chart API</h2>
<img id=google src="">
<p id=googleSrc class=tt></p>
```

Für die Textausgabe verwenden wir zusätzlich das in Abschnitt 3.3.5, Berechnungen mit »output«, vorgestellte `output`-Element. Als Erstes wird in JavaScript aber das aktuelle Datum erzeugt und das `meter`-Element initialisiert:

```
var today = new Date();
var m = document.getElementById("m");
m.min = new Date(today.getFullYear(), 0, 1);
m.max = new Date(today.getFullYear(), 11, 31);
// m.optimum = m.min-m.max/2;
m.value = today;
```

Die Variable `today` enthält die Anzahl an Millisekunden seit dem Beginn der UNIX-Epoche (dem 1.1.1970). Damit unser `meter`-Element eine vernünftige Skala erhält, wird der `min`-Wert auf den 1. Januar des aktuellen Jahres eingestellt, der `max`-Wert entsprechend auf den 31. Dezember. Der Wert des `meter`-Elements wird in der letzten Zeile des Listings eingestellt, und die grafische Anzeige ist komplett. Wer den hier ausgeklammerten `optimum`-Wert aktiviert (in diesem Fall die Mitte des Jahres), wird je nachdem, ob das Script in der ersten oder in der zweiten Jahreshälfte aufgerufen wird, eine entsprechende Veränderung der Anzeige erkennen. Wunderbar, wie einfach das neue Element zu verwenden ist.

Doch kommen wir nun zu den restlichen Elementen auf unserer HTML-Seite. Das mit der ID `op` gekennzeichnete `output`-Element wollen wir mit dem Prozentwert der vergangenen Tage belegen. Die Prozentrechnung wird mit `Math.round()` von ihren Kommastellen befreit, eine Genauigkeit, die für unser Beispiel ausreichend ist. Anschließend wird der `span`-Bereich (`opText`) mit diesem Wert belegt.

```
var op = document.getElementById("op");
op.value =
  Math.round(100/(m.max-m.min)*(m.value-m.min));
var opText = document.getElementById("opText");
opText.innerHTML = op.value;
var innerDIV = document.getElementById("innerDIV");
innerDIV.style.width=op.value+"%";
innerDIV.style.background = "green";
```

Der Rest dieses Beispiels hat zwar nichts mehr mit neuen HTML5-Techniken zu tun, wird aber aus Gründen der Vollständigkeit auch noch erklärt. Die verschachtelten `div`-Elemente wollen wir ebenfalls mit dem Prozentwert befüllen. Die Idee dahinter ist simpel: Ein erster `div`-Bereich wird mit einer fixen Breite in HTML definiert (hier 150px). Ein darin verschachteltes `div`-Element wird mit



der Breite von der berechneten Prozentzahl angegebenen und mit grüner Hintergrundfarbe gefüllt – ein einfacher Trick mit guter Wirkung. Abschließend wollen wir noch die Google Chart API mit einbeziehen. Beim Aufruf des Webservice müssen die Größe der Grafik (`chs`, hier `200x125` Pixel), der Typ der Grafik (`cht`, hier `gom`, *Google-O-Meter*) und die darzustellenden Daten (`chd`, hier der Prozentwert `op.value`) übergeben werden:

```
var google = document.getElementById("google");
google.src = "http://chart.apis.google.com/chart?chs=200x125&cht=gom&
    chd=t:"+op.value;
var gSrc = document.getElementById("googleSrc");
gSrc.innerHTML = google.src;
```

### 3.3.2 Fortschrittsanzeige mit »progress«

`progress` funktioniert ähnlich wie das eben vorgestellte `meter`-Element, mit dem Unterschied, dass es den Fortschritt eines laufenden Prozesses darstellt. Mögliche Prozesse sind ein Datei-Upload, den der Benutzer auslöst, oder der Download von externen Bibliotheken, wenn eine Applikation diese benötigt.

Für ein kurzes Beispiel wollen wir aber keine Dateien hochladen oder große Datenmengen herunterladen, es reicht, wenn wir uns selbst eine Aufgabe stellen und diese zu 100 Prozent erfüllen. Im Folgenden werden zehn Eingabelemente vom Typ `checkbox` definiert, und sobald alle aktiviert sind, soll der Fortschrittsbalken 100 % anzeigen.

```
<h1>Bitte aktivieren Sie alle Checkboxes</h1>
<form method=get>
  <input type=checkbox onchange=updateProgress()>
  <input type=checkbox onchange=updateProgress()>
<!-- und 8 weitere -->
  <p>
    Fortschritt: <progress value=0 max=10 id=pb></progress>
</form>
```

Das `progress`-Element wird mit einem Wert von 0 und einem Maximalwert von 10 initialisiert. Sobald ein Eingabelement aktiviert wird, ruft es die Funktion `updateProgress()` auf, die wie folgt aussieht:

```
function updateProgress() {
  var pb = document.getElementById("pb");
  var ip = document.getElementsByTagName("input");
  var cnt = 0;
  for(var i=0; i<ip.length; i++) {
```

```

    if (ip[i].checked == true) {
        cnt++;
    }
}
pb.value = cnt;
}

```

Die Variable `ip` enthält eine *NodeList* mit allen `input`-Elementen. Jedes dieser Elemente wird in der `for`-Schleife auf seinen Zustand überprüft. Sollte dieser aktiviert sein (`checked == true`), so erhöht sich die Zählervariable `cnt` um den Wert 1. Abschließend wird der Wert des `progress`-Elements auf den Wert der Zählervariable gestellt.

### 3.3.3 Auswahllisten mit »datalist«

Eine sehr häufig gewünschte neue Funktion für Formulare ist ein Aufklappmenü, das um eigene Einträge erweitert werden kann. Da das altbekannte `select`-Element auf die als `option`-Elemente angegebenen Werte beschränkt ist, ersannen Webentwickler verschiedene JavaScript-Kunstgriffe, durch die Textfelder um eine erweiterbare Auswahlliste ergänzt werden können.

Die HTML5-Spezifikation beinhaltet eine sehr elegante Lösung für dieses Problem. Das neue `datalist`-Element wurde so definiert, dass es als Container für das schon bekannte `option`-Element dient. Jedem `input`-Element kann nun ein `datalist`-Element zugewiesen werden, das bei Bedarf die Auswahlmöglichkeiten anzeigt. Browser, die das `datalist`-Element nicht unterstützen, zeigen nur das leere Textfeld an.

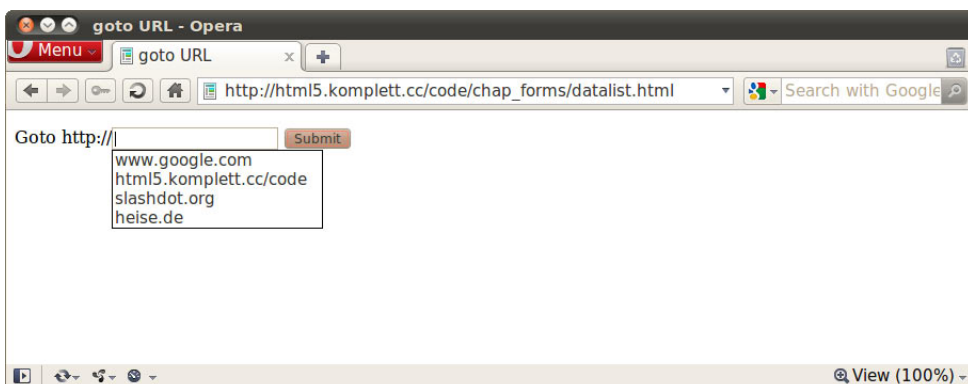


Abbildung 3.7: Opera mit der Darstellung eines »datalist«-Elements

Listing 3.1 zeigt die Verwendung des neuen Elements. Das `input`-Element wird vom Typ `text` definiert, und das Attribut `list` verweist auf die `id` des `datalist`-Elements (in diesem Fall `homepages`). Das `autofocus`-Attribut positioniert die Schreibmarke beim Laden der Seite automatisch innerhalb des Textfeldes (vergleiche Abschnitt 3.2.1, Fokussieren mit »`autofocus`«) und sorgt, zumindest beim Browser Opera, dafür, dass sich die Auswahlliste öffnet.

Für die `option`-Elemente innerhalb der `datalist` ist es ausreichend, das `value`-Attribut zu befüllen. Weitere Attribute und ein Text-Node sind zwar möglich, werden aber bei dieser Verwendung nicht benötigt. Beim Anklicken der `SUBMIT`-Schaltfläche wird dem Inhalt des Textfeldes die Zeichenkette `http://` vorangestellt und der Browser an die so entstandene URL umgeleitet (`window.location`).

```
<form>
  <p>
    <label for=url>Goto</label>
    http://<input type=text id=url name=homepage
      list=hompages autofocus>
  <datalist id=hompages>
    <option value=www.google.com>
    <option value=html5.komplett.cc/code>
    <option value=slashdot.org>
    <option value=heise.de>
  </datalist>
  <input type=submit
    onclick="window.location =
      'http://' + document.getElementById('url').value;
    return false;" >
</form>
```

**Listing 3.1:** Das »`datalist`«-Element, gefüllt mit Internet-Adressen

Wenn Sie ältere Browser ebenfalls mit einer Auswahlliste ausstatten möchten, ohne den HTML-Code zu duplizieren, können Sie auf folgenden Trick zurückgreifen. Da Browser, die das `datalist`-Element unterstützen, ein eingeschlossenes `select`-Element ignorieren, zeigen sie das neue HTML5-Auswahlelement an. Ältere Browser hingegen zeigen zu dem Textfeld eine Auswahlliste mit vorgegebenen Links an, die bei einer Änderung der Auswahl in das Textfeld eingefügt werden.

```
<datalist id=hompages>
<select name=homepage
  onchange="document.getElementById('url').value =
```

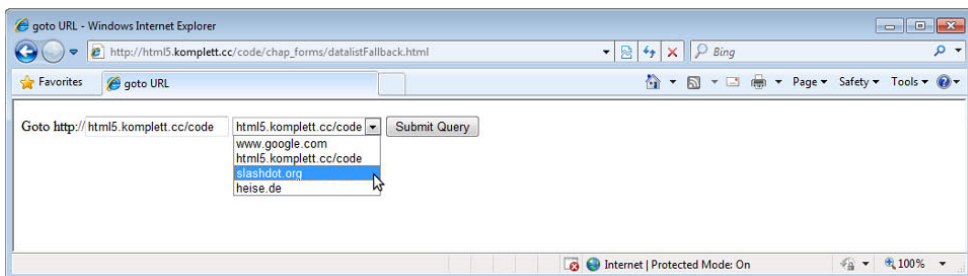
```

    document.forms[0].homepage[1].value" >
<option value=www.google.com>www.google.com
<option
value=html5.komplett.cc/code>html5.komplett.cc/code
<option value=slashdot.org>slashdot.org
<option value=heise.de>heise.de
</select>
</datalist>

```

**Listing 3.2:** Eine »datalist« mit dem Fallback für ältere Browser

Wie in diesem Listing zu sehen ist, müssen die `option`-Elemente mit einem Text-Node versehen werden, da das »alte« `select`-Element nicht den Inhalt des `value`-Attributs anzeigt, sondern den Text. Das `onchange`-Event in dem `select`-Element setzt den aktuellen Text des Auswahlmenüs in das Textfeld ein (vergleiche Abbildung 3.8).



**Abbildung 3.8:** Eine Kombination aus »input«- und »select«-Elementen als Fallback für ältere Browser (hier Internet Explorer 8)

### 3.3.4 Kryptografische Schlüssel mit »keygen«

Das `keygen`-Element hat bereits eine lange Geschichte im Browser Mozilla Firefox (enthalten seit Version 1.0), trotzdem meldete Microsoft große Bedenken bei der Implementierung in HTML5 an. `keygen` wird zur Erzeugung von kryptografischen Schlüsseln verwendet, und so kompliziert das klingt, so kompliziert ist es leider auch.

Ganz einfach gesprochen ist die Idee hinter diesem Element folgende: Der Browser erzeugt ein Schlüsselpaar, das aus einem öffentlichen Schlüssel (*public key*) und einem privaten Schlüssel (*private key*) besteht. Der öffentliche Schlüssel wird mit den anderen Formulardaten verschickt und steht anschließend der Server-Anwendung zur Verfügung, während der private Schlüssel im Browser gespeichert bleibt. Nach diesem Schlüsselaustausch haben Ser-

ver und Browser die Möglichkeit, verschlüsselt zu kommunizieren, und zwar ohne SSL-Zertifikate. Das klingt nach einer praktischen Lösung für die lästigen selbst signierten Zertifikate, die die Browser immer beanstanden müssen, ist es aber leider nicht, denn die Identität des Servers kann nur aufgrund eines Zertifikats gewährleistet werden, das von einer vertrauenswürdigen Zertifizierungsstelle (der *Certificate Authority, CA*) unterschrieben worden ist.

Da `keygen` SSL nicht ablösen kann, wofür soll das neue Element dann verwendet werden? Wie die Dokumentation von Mozilla erklärt, hilft das `keygen`-Element, ein Zertifikat zu erstellen, das vom Server unterschrieben werden kann (*signed certificate*). Um diesen Schritt ganz sicher zu gestalten, ist es normalerweise notwendig, dass der Antragsteller persönlich bei der Behörde erscheint. Da das Ausstellen von signierten Zertifikaten eher etwas für Experten ist, werden wir die Beschreibung dieses Elements und seiner Attribute eher kurz halten.

Folgendes kurze HTML-Dokument erzeugt eine `keygen`-Schaltfläche:

```
<!DOCTYPE html>
<meta charset="utf-8">
<title>keygen Demo</title>
<form method=post action=submit.html>
  <keygen id=kg challenge=hereismychallenge name=kg>
  <input type=submit>
</form>
```

Außer den bekannten Attributen wie `autofocus`, `disabled`, `name` und `form` besitzt das `keygen`-Element zwei spezielle Attribute: `keytype` und `challenge`. Vor allem `keytype` ist interessant, da der Browser anhand dieses Eintrags entscheidet, ob er die Funktion dieses Elements unterstützt. Momentan gibt es nur einen gültigen `keytype`, und zwar `rsa`, ein kryptografisches System, das im Jahr 1977 am *Massachusetts Institute of Technology (MIT)* entwickelt wurde. Wird kein `keytype` angegeben (wie im vorangegangenen Beispiel), wird als Standardwert `rsa` verwendet. Die Spezifikation sieht auch vor, dass ein Browser überhaupt keinen `keytype` unterstützen muss, was wohl auf das Veto von Microsoft gegen das Element zurückgeht. Das optionale `challenge`-Attribut erhöht die Sicherheit beim Schlüsselaustausch. Für weiterführende Informationen verwenden Sie bitte die Links am Ende dieses Abschnitts.

Unterstützt der Browser die RSA-Schlüsselerzeugung, so kann er dem Benutzer eine Auswahlliste für die Länge und damit die Sicherheit des Schlüssels anbieten (vergleiche Abbildung 3.9).

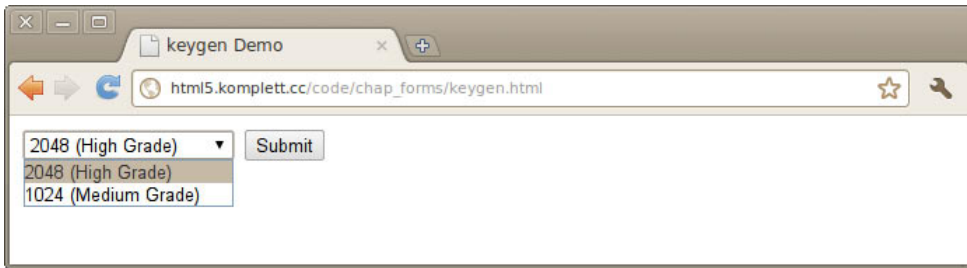


Abbildung 3.9: Auswahl der Schlüssellänge in Google Chrome

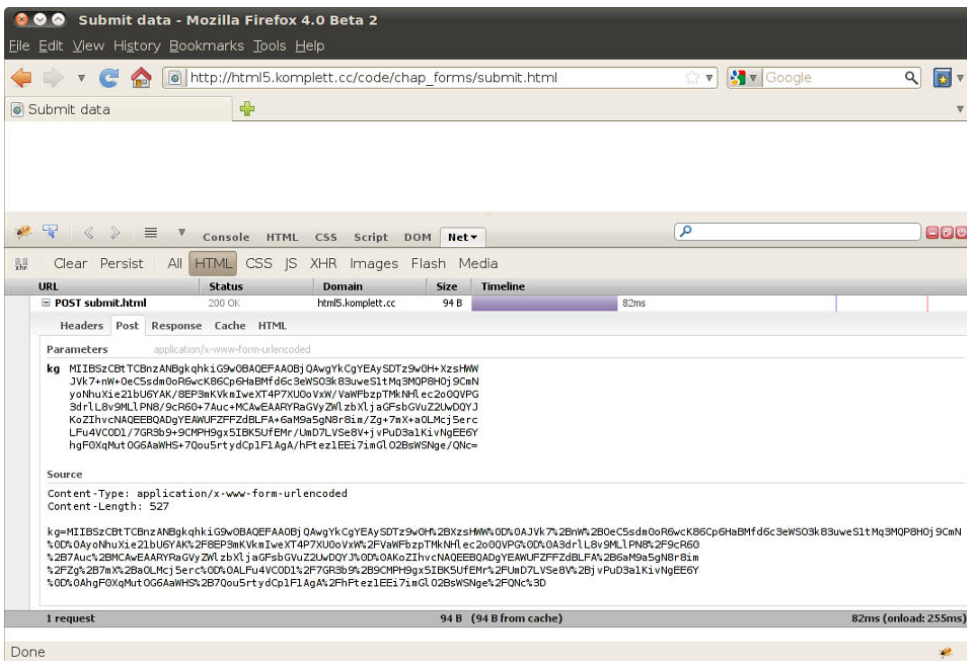


Abbildung 3.10: Der öffentliche Schlüssel des »keygen«-Elements, dargestellt in Firebug

Das Resultat nach dem Abschicken dieses Formulars zeigt Abbildung 3.10: Die POST-Variablen enthält den zur Verschlüsselung notwendigen öffentlichen Schlüssel (hier dargestellt in dem äußerst hilfreichen Firefox-Add-On Firebug).

Wenn Sie bisher noch wenig Kontakt mit Kryptografie hatten, sich aber dafür interessieren, bietet, wie so oft, Wikipedia eine gute Einstiegslektüre:

<http://de.wikipedia.org/wiki/Public-Key-Infrastruktur>

<http://de.wikipedia.org/wiki/Challenge-Response-Authentifizierung>

HINWEIS



### 3.3.5 Berechnungen mit »output«

»Das `output`-Element enthält das Ergebnis einer Berechnung«. So lautet die sehr knappe Erklärung in der HTML5-Spezifikation, und genau das findet man auch auf den meisten Internet-Seiten, die das neue Element beschreiben. Das klingt sehr vernünftig, aber was für eine Art von Berechnung ist damit gemeint? Wieso braucht es dazu ein eigenes Element?

In der Regel handelt es sich dabei um Berechnungen, die aus Eingabefeldern auf einer Webseite zustande kommen. Ein Beispiel, das vielleicht allen geläufig ist, wäre ein elektronischer Einkaufswagen, in dem die Stückzahl für die Produkte in einem `input`-Feld eingestellt werden kann. Mithilfe des optionalen `for`-Attributs lässt sich festlegen, welche Felder in die Berechnung mit einfließen. Dabei werden ein oder mehrere `id`-Attribute anderer Felder des Dokuments referenziert.

Um das `output`-Element auszuprobieren, wollen wir so einen kleinen Einkaufswagen programmieren, in dem drei verschiedene Produkte vorhanden sind. Die Stückzahl jedes dieser Produkte kann mithilfe eines Eingabefeldes verändert werden. Gleichzeitig wird unter dem Einkaufswagen die Anzahl der Waren und die Gesamtsumme angezeigt. Abbildung 3.11 zeigt einen Warenkorb mit fünf Einträgen.

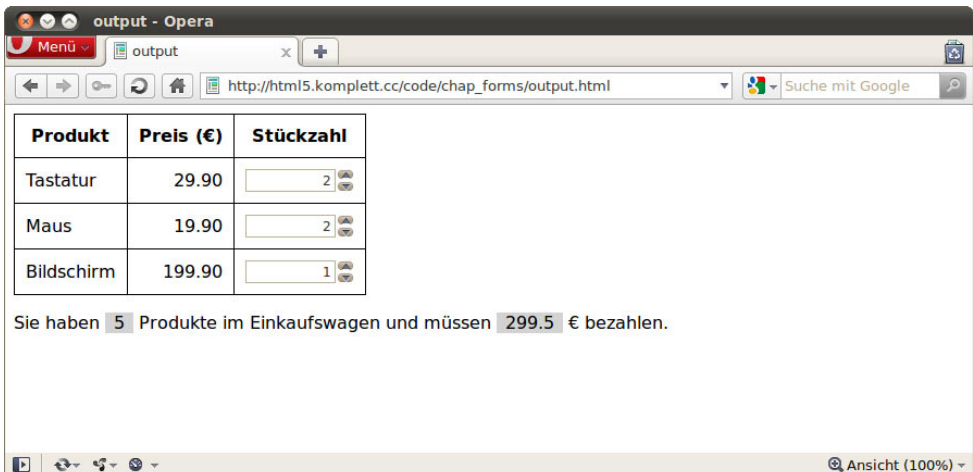


Abbildung 3.11: Zwei »output«-Elemente zeigen die Anzahl der Produkte und deren Gesamtsumme an.

Der Code für das Beispiel ist schnell erklärt: Um bei jeder Änderung der Stückzahl die output-Elemente zu aktualisieren, verwenden wir das `oninput`-Event des Formulars:

```
<form oninput="updateSum();">
  <table>
    <tr><th>Produkt<th>Preis (€)<th>Stückzahl
    <tr><td>Tastatur<td class=num id=i1Price>29.90<td>
    <input name=i1 id=i1 type=number min=0 value=0 max=99>
    <tr><td>Maus<td class=num id=i2Price>19.90<td>
```

Die output-Elemente sind im Anschluss an die Tabelle mit den Produkten definiert und verweisen über das `for`-Attribut auf die IDs der input-Felder:

```
<p>Sie haben <output name=sumProd for="i1 i2 i3"
  id=sumProd></output> Produkte im Einkaufswagen und
  müssen <output name=sum for="i1 i2 i3" id=sum></output>
  € bezahlen.
```

Im JavaScript-Code läuft eine Schleife über alle `input`-Elemente. Sie zählt die Stückzahlen zusammen und errechnet den Gesamtpreis.

```
function updateSum() {
  var ips = document.getElementsByTagName("input");
  var sum = 0;
  var prods = 0;
  for (var i=0; i<ips.length; i++) {
    var cnt=Number(ips[i].value);
    if (cnt > 0) {
      sum += cnt * Number(document.getElementById(
        ips[i].name+"Price").innerHTML);
      prods += cnt;
    }
  }
  document.getElementById("sumProd").value = prods;
  document.getElementById("sum").value = sum;
}
```

Den Preis des Produkts holen wir uns direkt aus der Tabelle. Dabei verwenden wir den `innerHTML`-Wert der entsprechenden Tabellenspalte und wandeln diesen mit der JavaScript-Funktion `Number()` in eine Zahl um. Gleiches gilt auch für den Wert im `input`-Feld (`ips[i].value`), denn ohne diese Umwandlung würde JavaScript die Zeichenketten addieren, was nicht zu dem gewünschten Ergebnis führt. Abschließend werden die errechneten Werte in die `value`-Attribute der `output`-Elemente eingesetzt.



## 3.4 Clientseitige Formular-Überprüfung

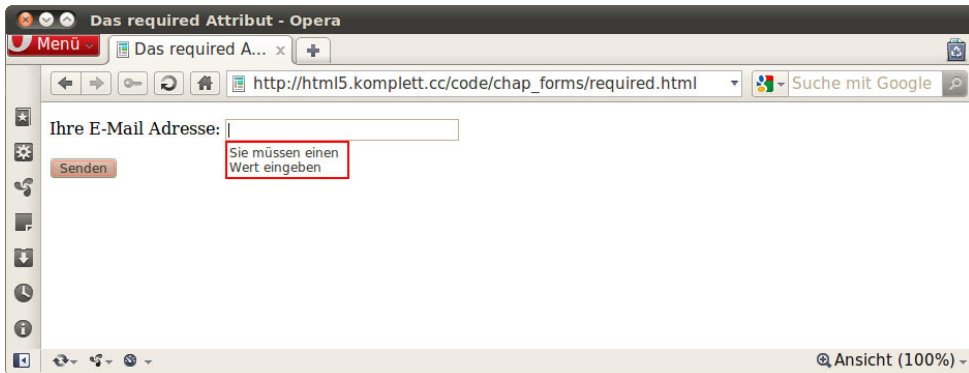
Ein Vorteil der neuen Elemente und Attribute bei Formularen ist, dass die Eingabe für Benutzer erleichtert wird (zum Beispiel wird ein Kalender zum Eingeben eines Datums angeboten). Ein weiterer großer Vorteil ist die Möglichkeit, den Formular-Inhalt bereits vor dem Abschicken überprüfen zu können und den Benutzer auf mögliche Fehler hinzuweisen. Jetzt werden Sie vielleicht sagen, dass das ein alter Hut ist, denn diese Form der Überprüfung kennt man bereits seit vielen Jahren. Das stimmt, aber bisher musste dieser Schritt immer mithilfe von selbst programmiertem JavaScript-Code erledigt werden. Durch jQuery und ähnliche Bibliotheken wurde diese Aufgabe zwar deutlich erleichtert und der Code wartbarer, aber es bleibt die Abhängigkeit von einer externen Bibliothek.

Mit HTML5 ändert sich das grundlegend: Sie definieren die Vorgaben für die Eingabefelder in HTML, und der Browser überprüft, ob die Felder korrekt ausgefüllt wurden. Das ist ein großer Schritt vorwärts, der viele redundante Zeilen JavaScript-Code unnötig macht. Ein Minimalbeispiel wird Sie überzeugen:

```
<form method=get action=required.html>
  <p><label>Ihre E-Mail-Adresse:
    <input type=email name=email required></label>
  <p><input type=submit>
</form>
```

Was passiert, wenn Sie das Formular in dem oben abgedruckten Listing ohne die Angabe einer E-Mail-Adresse abschicken, sehen Sie in Abbildung 3.12. Opera zeigt die Fehlermeldung: *Sie müssen einen Wert eingeben*. Wenn Sie die Opera-Benutzeroberfläche auf eine andere Sprache eingestellt haben, so erscheint diese Meldung in der entsprechenden Sprache. Natürlich kann man diese Fehlermeldungen auch noch mit JavaScript anpassen, mehr dazu erfahren Sie etwas später.

Damit aber noch nicht genug: Da das Feld vom Typ `email` definiert ist, meldet Opera auch einen Fehler, wenn keine gültige E-Mail-Adresse eingegeben wurde (vergleiche Abbildung 3.13).



**Abbildung 3.12:** Die Fehlermeldung bei einem leeren Eingabefeld mit dem Attribut »required« (Opera)



**Abbildung 3.13:** Die Fehlermeldung von Opera beim Eingeben einer ungültigen E-Mail-Adresse

WebKit-basierte Browser wie Google Chrome oder Safari unterstützen aktuell zwar die Überprüfung, geben aber keine Fehlermeldung aus. Sie umrahmen das ungültige Feld und positionieren den Cursor innerhalb des Feldes, um zumindest anzuzeigen, dass irgendetwas nicht stimmt.

Bei aller Euphorie über die clientseitige Überprüfung von Formular-Eingaben dürfen Sie nicht vergessen, dass dieser Schritt die serverseitige Kontrolle nicht überflüssig machen kann. Ein potenzieller Angreifer kann diese Mechanismen mit wenig technischem Aufwand umgehen.

**HINWEIS**



### 3.4.1 Das »invalid«-Event

Bei der Überprüfung des Formulars wird für Elemente, die einen ungültigen Inhalt haben, das Event *invalid* ausgelöst. Das können wir uns zunutze machen und individuell auf fehlerhafte Werte reagieren.

```
window.onload = function() {
  var inputs = document.getElementsByTagName("input");
  for (var i=0; i<inputs.length; i++) {
    inputs[i].addEventListener("invalid", function() {
      alert("Feld "+this.labels[0].innerHTML
        +" ist ungültig");
      this.style.border = 'dotted 2px red';
    }, false);
  }
}
```

Nachdem die Seite geladen ist, wird (wie schon im Beispiel auf Seite 75) eine Liste aller `input`-Elemente generiert. An jedes Element wird anschließend ein Event-Listener angehängt, der den Fehlerfall behandelt. Im vorliegenden Beispiel wird ein `alert`-Fenster geöffnet, und das Element erhält einen rot gepunkteten Rahmen. Für den Text im `alert`-Fenster wird die Beschriftung des `input`-Elements verwendet.

Bei Formularen mit vielen Eingabefeldern ist diese Vorgehensweise nicht ideal. Der Anwender muss für jede fehlerhafte Eingabe die OK-Schaltfläche anklicken und anschließend im Formular das betreffende Feld suchen und erneut ausfüllen. Manchmal wäre es günstiger, wenn der Anwender sofort nach dem Ausfüllen eine Benachrichtigung bekäme, sollte das Feld einen ungültigen Inhalt enthalten. Das wollen wir im nächsten Abschnitt probieren.

### 3.4.2 Die »checkValidity«-Funktion

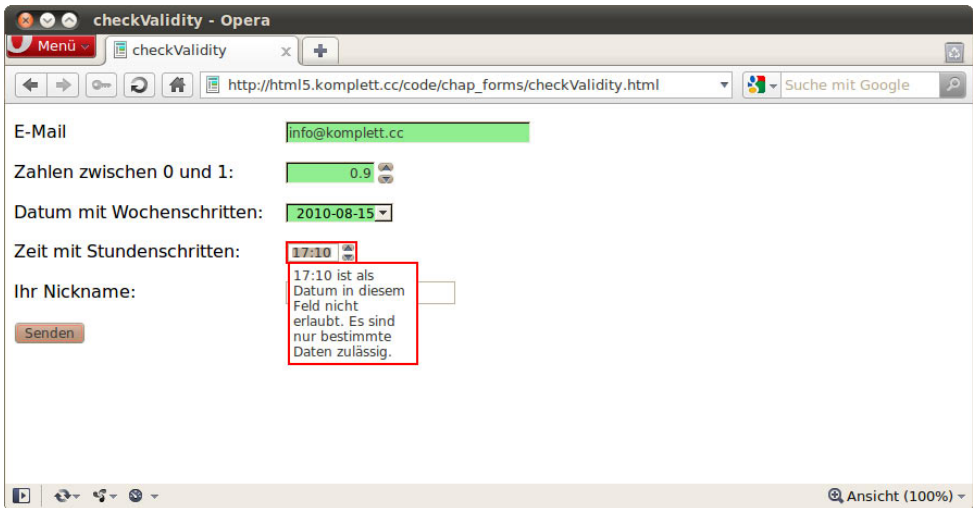
Um die Überprüfung eines `input`-Elements auszulösen, wird die `checkValidity`-Funktion für dieses Element aufgerufen. Was normalerweise passiert, wenn das Formular abgeschickt wird, kann man aber auch »von Hand« starten:

```
<input type=email name=email
  onchange="this.checkValidity();">
```

Gibt man eine ungültige E-Mail-Adresse ein und verlässt man das Eingabefeld (entweder mit der Tabulator-Taste oder durch einen Mausklick auf eine andere Stelle im Browser), so meldet der Browser (zurzeit zumindest Opera) den Fehler unmittelbar (vergleiche Abbildung 3.13). Noch eleganter wird die Fehlerbehandlung, wenn wir an das `onchange`-Event von allen `input`-Elementen eine Funktion zum Überprüfen der Eingabe hängen.

```
window.onload = function() {
  var inputs = document.getElementsByTagName("input");
  for (var i=0; i<inputs.length; i++) {
    if (!inputs[i].willValidate) {
      continue;
    }
    inputs[i].onchange = function() {
      if (!this.checkValidity()) {
        this.style.border = 'solid 2px red';
        this.style.background = '';
      } else {
        this.style.border = '';
        this.style.background = 'lightgreen';
      }
    }
  }
}
```

In der bereits bekannten Schleife über alle `input`-Elemente wird als Erstes kontrolliert, ob das Element für eine Überprüfung zur Verfügung steht. Enthält `willValidate` nicht den Wert `true`, wird die Schleife mit dem nächsten Element fortgesetzt. Andernfalls wird das `onchange`-Event mit einer anonymen Funktion belegt, in der die `checkValidity`-Funktion aufgerufen wird. `this` bezieht sich innerhalb der anonymen Funktion auf das `input`-Element. Schlägt die Gültigkeitsprüfung fehl, so wird das Element mit einer roten Umrandung versehen; im anderen Fall wird der Hintergrund des Elements hellgrün eingefärbt. Das Zurücksetzen der Hintergrundfarbe beziehungsweise des Rahmens auf eine leere Zeichenkette ist notwendig, damit der Browser bei einer richtigen Eingabe nach einer falschen Eingabe die Formatierung wieder auf den Standardwert stellt. Abbildung 3.1 zeigt, wie die `checkValidity`-Funktion einen Fehler bei der Zeiteingabe anmahnt.



**Abbildung 3.14:** Opera zeigt die Fehlermeldung für eine inkorrekte Zeiteingabe (in diesem Fall eine Verletzung des »step«-Attributs)

Wenn Sie die Fehlerbehandlung lieber noch interaktiver gestalten möchten, können Sie statt des `onchange`-Events auch das in HTML5 neue `oninput`-Event verwenden. Anders als `onchange`, das beim Verlassen des Feldes gestartet wird, kommt `oninput` nach jedem veränderten Zeichen zum Einsatz. Was bisher etwas mühsam mithilfe der Tastatur-Events `keyup` beziehungsweise `keydown` programmiert wurde, übernimmt jetzt das `oninput`-Event. Ein weiterer Vorteil von `oninput` ist, dass der Event-Listener nur einmal an das ganze Formular angehängt werden muss und nicht an jedes einzelne `input`-Element. Für das vorangegangene Beispiel könnte man damit auf den gesamten JavaScript-Code verzichten und die Formular-Definition wie folgt ändern:

```
<form method=get oninput="this.checkValidity();"
  action=checkValidity.html >
```

Man verzichtet damit zwar auf das Verändern von Rahmen und Hintergrundfarbe, verkürzt aber auch den Quelltext deutlich. Das unmittelbare Reagieren auf einen Tastendruck kann in manchen Fällen sehr hilfreich sein, beim Ausfüllen eines Formularfelds reicht es aber meist, wenn der Inhalt erst dann überprüft wird, wenn das Feld vollständig ausgefüllt wurde.

### 3.4.3 Fehlerbehandlung mit »setCustomValidity()«

Wenn Ihnen all die bisher vorgestellten Möglichkeiten zur Fehlerbehandlung noch nicht ausreichend erscheinen, können Sie sich auch selbst eine Funktion zur Überprüfung des Inhalts programmieren. Im folgenden Beispiel wird ein Eingabefeld vom Typ `email` definiert, wodurch der Browser schon die Überprüfung der gültigen E-Mail-Adresse übernimmt. Zusätzlich möchten wir aber noch drei E-Mail-Domains ausschließen.

```
var invalidMailDomains = [
    'hotmail.com', 'gmx.com', 'gmail.com' ];

function checkMailDomain(item) {
    for (var i=0; i<invalidMailDomains.length; i++) {
        if (item.value.match(invalidMailDomains[i]+'$')) {
            item.setCustomValidity('E-Mail-Adressen von '
                +invalidMailDomains[i]+' sind nicht erlaubt.');
```

Jedes Element im Array `invalidMailDomains` wird mit dem Wert des `input`-Elements verglichen. Die JavaScript-Funktion `match()` arbeitet mit regulären Ausdrücken, weshalb wir an den Domain-Namen noch ein `$`-Zeichen anhängen, das das Ende der Zeichenkette spezifiziert. Stimmen die Zeichenketten überein, so wird die `setCustomValidity`-Funktion aufgerufen und ihr die entsprechende Fehlermeldung übergeben. Handelt es sich nicht um einen Domain-Namen aus dem Array, wird `setCustomValidity()` mit einer leeren Zeichenkette aufgerufen. Intern wird dadurch die Variable `validationMessage` an das `input`-Element angehängt, die Opera anschließend auch korrekt anzeigt (vergleiche Abbildung 3.15). Der abschließende Aufruf der `checkValidity`-Funktion löst die Überprüfung aus und führt zu der eben erwähnten Fehlermeldung.

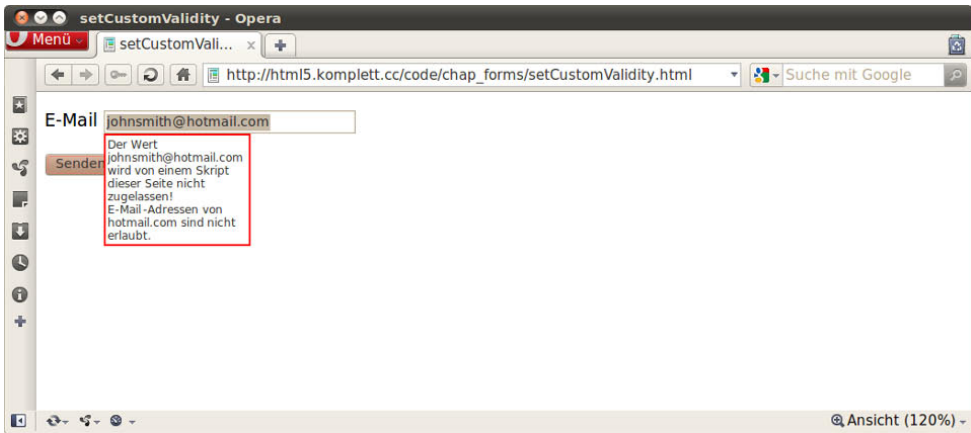


Abbildung 3.15: Opera zeigt die Fehlermeldung bei einer manuellen Fehlerbehandlung (Überprüfung der E-Mail-Domain)

### 3.4.4 Zusammenfassung der Gültigkeitsprüfungen

Tabelle 3.3 zeigt eine Auflistung aller `input`-Attribute beziehungsweise Validierungsfunktionen, die bei der Gültigkeitsüberprüfung zur Verfügung stehen, und die Szenarien, in denen sie auftreten.

Attribut/Funktion	Problem
<code>required</code>	Es wurde kein Wert für das Feld eingegeben.
<code>type=email, url</code>	Der eingegebene Wert entspricht nicht dem verlangten Typ.
<code>pattern</code>	Der eingegebene Wert entspricht nicht dem geforderten Muster.
<code>maxLength</code>	Der eingegebene Wert ist länger als erlaubt.
<code>min, max</code>	Der eingegebene Wert ist zu klein bzw. zu groß.
<code>step</code>	Die verlangte Schrittweite beim eingegebenen Wert wurde nicht eingehalten.
<code>setCustomValidity()</code>	Die zusätzlich aufgestellten Kriterien für dieses Feld wurden nicht erfüllt.

Table 3.3: Fehlermöglichkeiten bei der Gültigkeitsprüfung von Formularfeldern

### 3.4.5 Oder doch nicht prüfen? »formnovalidate«

Nun, da wir uns so ausführlich mit der Fehlerbehandlung beschäftigt haben, folgt die Erklärung, wie man sich an all den Regeln vorbeimogeln kann: mit dem Attribut `formnovalidate`. Im ersten Moment erscheint es vielleicht ein bisschen merkwürdig, all die mühevoll definierten Regeln einfach so beiseite zu lassen und das Formular auch ohne eine Prüfung abzuschicken. Die Spezifikation enthält dazu eine kurze Erklärung, die das Rätsel schnell löst. Der typische Anwendungsfall für das Überspringen der Prüfung ist ein Formular, das der Anwender nicht auf einmal ausfüllen kann oder will. Dadurch, dass man das `formnovalidate`-Attribut einer `submit`-Schaltfläche hinzufügt, kann der bisher eingegebene Inhalt zwischengespeichert werden.

Beim Abschicken des Formulars mit `formnovalidate` werden die bereits ausgefüllten Felder an den Server gesendet. Um ein mögliches Zwischenspeichern muss sich die Server-Anwendung kümmern.

HINWEIS



Stellen Sie sich vor, Sie füllen ein Support-Formular für Ihre defekte Digitalkamera aus. Nachdem Sie ausführlich alle Angaben zu dem aufgetretenen Fehler gemacht haben, wird auf der Internet-Seite nach der Seriennummer der Kamera gefragt. Da die Sie die Kamera aber gerade nicht zur Hand haben und die mühevoll eingegebenen Informationen nicht verlieren möchten, klicken Sie auf die `ZWISCHENSPEICHERN`-Schaltfläche und können sich in Ruhe auf die Suche nach der Kamera begeben. Diese Schaltfläche wird wie folgt definiert:

```
<p><input type=submit formnovalidate
  value="Zwischenspeichern" name=save id=save>
```

Im abschließenden Beispiel wird die Idee mit dem Support-Formular vollständig ausgearbeitet.

## 3.5 Beispiel: Ein Support-Formular

In diesem Beispiel werden die bisher vorgestellten neuen Elemente und Attribute in einem Formular verwendet. Das Formular könnte, in einer erweiterten Form, auf der Webseite eines Elektronik-Verkäufers Verwendung finden.

Zu Beginn werden persönliche Informationen vom Klienten abgefragt (in diesem Beispiel nur der Name, eine E-Mail-Adresse, eine Telefon- und eine Faxnummer). Der zweite Teil des Formulars betrifft die technischen Daten und den



## Kapitel 3 Intelligente Formulare

Defekt des Geräts. Im untersten Teil der Webseite wird ein Fortschrittsbalken angezeigt, der den Anwender aufmuntern soll, das Formular fertig auszufüllen (vergleiche Abbildung 3.16).

The screenshot shows a web browser window with the address bar displaying "html5.komplett.cc/code/chap\_forms/support.html". The form contains the following fields and content:

- E-Mail:** info@html5.komplett.cc
- Faxnummer:** (empty)
- Technische Informationen:**
  - Datum des Defekts:** 2010-11-11
  - Fehlermeldung:** Lens Error. Camera restart required.
  - Problembeschreibung:** Nach dem Einschalten des Fotoapparats öffnet sich die Abdeckung der Linse nicht und die oben beschriebene Fehlermeldung erscheint auf der Rückseite der Kamera. In weiterer Folge schaltet sich die Kamera automatisch aus.
  - Kommentar:** Das Gerät ist erst drei Wochen alt und wurde immer sehr schonend behandelt.
  - Gerät:** Digitalkamera
  - Seriennummer:** (empty)
- Fortschritt:** A progress bar showing approximately 25% completion.
- Buttons:** "Zwischenspeichern [Z]" and "Abschicken [A]"

**Abbildung 3.16:** Das fast fertig ausgefüllte Support-Formular

Der HTML-Code für das Formular beginnt mit dem Laden einer externen JavaScript-Datei und dem bereits bekannten Aufruf `window.onload`.

```
<script src="support.js"></script>
<script>
  window.onload = function() {
    initEventListener();
  }
</script>
```

Die `initEventListener`-Funktion läuft über alle `input`-Elemente und belegt das `onchange`-Event mit einer anonymen Funktion, die das entsprechende Element auf seine Gültigkeit überprüft.

```
function initEventListener() {
  var inputs = document.getElementsByTagName("input");
  for (var i=0; i<inputs.length; i++) {
    if (!inputs[i].willValidate) {
      continue;
    }
    inputs[i].onchange = function() {
      this.checkValidity();
    }
  }
}
```

Der Event-Listener wird nur dann angehängt, wenn das Element eine Möglichkeit der Überprüfung hat. Im vorliegenden Beispiel haben die beiden Schaltflächen zum Absenden beziehungsweise zum Zwischenspeichern keine Überprüfungsmöglichkeit und bekommen daher kein `onchange`-Event. Wie im vorangegangenen Abschnitt schon erklärt wurde, ist die Überprüfung der einzelnen Formularfelder nach dem Ausfüllen des Feldes angenehmer als die Überprüfung des gesamten Formulars mit dem `oninput`-Event.

Um die Benutzerfreundlichkeit des Formulars zu verbessern, wollen wir die als `required` markierten Elemente hervorheben, damit dem Anwender sofort klar wird, welches die wichtigen Felder sind. Glücklicherweise müssen wir dazu nicht jedes Element mit einem extra Stil versehen, CSS3 bringt den neuen Selektor `:required` mit, der genau für diesen Fall gedacht ist. Die folgende Anweisung rahmt alle vorgeschriebenen Elemente mit oranger Farbe ein.

```
:required { border-color: orange; border-style: solid; }
```

Die Definition der einzelnen `input`-Felder birgt keine großen Überraschungen. E-Mail-Adresse und Telefonnummer haben ihre eigenen Typen und sind vorgeschrieben; das Datum, an dem der Defekt auftrat, ist vom Typ `date` und kann daher mit einem Kalenderfenster ausgewählt werden. Das zweispaltige Layout im oberen Teil der Webseite wird mit `div`-Elementen erreicht, die nebeneinander liegen. Trotzdem möchten wir, dass Anwender, die die Tabulatortaste zum Weiterspringen verwenden, das Formular von oben nach unten ausfüllen und nicht, der HTML-Logik folgend, zuerst die linke und dann die rechte Spalte.

Erreicht wird das mithilfe des `tabindex`-Attributs, wodurch ein Drücken der Tabulatortaste in einem Feld den Cursor auf das Feld mit dem nächsthöheren `tabindex`-Wert stellt.

```
<div style="float:left">
<p><label>Ihr Name:
<input tabindex=1 type=text required autofocus
placeholder="Max Mustermann" name=name></label>
<p><label >E-Mail
<input tabindex=3 type=email name=email required></label>
</div>
<div style="float:left;margin-left:10px;">
<p><label>Telefonnummer
<input tabindex=2 type=tel name=tel required></label>
<p><label>Faxnummer
<input tabindex=4 type=tel name=fax></label>
</div>
```

Etwas spannender wird es bei den `textarea`-Feldern. Viel Neues brachte HTML5 für diesen Typ nicht, aber wie Sie in Abbildung 3.16 sehen können, enthält jedes Textfeld eine kleine grafische Anzeige oberhalb, die darstellt, wie viele Zeichen in dem Feld noch getippt werden können. Sicherlich haben Sie es gleich erkannt: Wir verwenden dazu das neue `meter`-Element, das uns bereits aus Abschnitt 3.3.1, Anzeigen von Messgrößen mit »meter«, vertraut ist.

```
<p><label>Fehlermeldung
<textarea placeholder="Lens . Camera restart."
name=errmsg required rows=5 cols=50
title="maximal 200 Zeichen">
</textarea></label><meter value=0 max=200
tabindex=-1></meter>
```

Das `meter`-Element wird mit einem Maximalwert von 200 initialisiert, exakt dem Wert, der im `title`-Attribut der `textarea` als Maximum angegeben ist. Gibt ein Anwender mehr als die vorgeschriebenen Zeichen ein, wird das `meter`-Element rot und warnt vor dem zu langen Text. Der Browser wird den zu langen Text aber trotzdem abschicken, da wir die `textarea` nicht limitiert haben. Es handelt sich hier also mehr um einen Hinweis, als um eine strikte Vorgabe. Die JavaScript-Funktion zum Aktualisieren der `meter`-Elemente lautet `updateTAMeters()` und wird für alle `textareas` ausgeführt:

```
function updateTAMeters() {
var textfs = document.getElementsByTagName("textarea");
for(var i=0; i<textfs.length; i++) {
```

```
    textfs[i].labels[0].nextSibling.value =  
        textfs[i].textLength;  
    }  
}
```

Der Vorteil der Schleife ist, dass wir nun beliebig viele `textarea`-Elemente hinzufügen können, und sofern sie ein `meter`-Element besitzen, werden diese automatisch aktualisiert. Um das zu erreichen, müssen wir zu einem Trick aus der DOM-Kiste greifen: Die in dem oben stehenden Listing fett gedruckte Zuweisung greift auf die DOM-Funktion `nextSibling` zu, einen Verweis auf das folgende Element. Führen wir uns zum besseren Verständnis noch einmal den HTML-Code für das Textfeld und den Status-Balken vor Augen. Das `textarea`-Element ist von einem `label`-Element eingeschlossen, auf das das gesuchte `meter`-Element folgt. Um vom `textarea`-Element auf das `meter`-Element zu kommen, verwenden wir die `labels`-Eigenschaft des Textfeldes. Dabei handelt es sich um ein `NodeList`-Array, von dem uns das erste Element (also das mit dem Index 0) interessiert, weil das darauffolgende Element (der `nextSibling`) das `meter`-Element ist.

Bei genauerer Betrachtung ist die Vorgehensweise also gar nicht so kompliziert, sie birgt aber ihre Tücken. Sollte sich zwischen dem abgeschlossenen `label`-Element und dem `meter`-Element ein Leerzeichen oder ein Zeilenumbruch verirren, dann funktioniert unsere Status-Anzeige nicht mehr. Der `nextSibling` ist dann nämlich ein Text-Element, und in der `for`-Schleife erreichen wir das `meter`-Element nicht mehr.

Als Nächstes wollen wir uns um die Fortschrittsanzeige am Ende des Formulars kümmern. Leicht zu erraten war, dass es sich dabei um ein `progress`-Element handelt, spannender wird, wie sich das Aktualisieren dieses Elements in JavaScript elegant ausdrücken lässt. Zuerst sehen Sie hier den HTML-Code für das Element:

```
<label>Fortschritt:  
  <progress id=formProgress value=0  
    tabindex=-1></progress></label>
```

Das `progress`-Element bekommt eine `id`, einen Anfangswert von 0 (`value`) und einen negativen `tabindex` zugewiesen, was dazu führt, dass das Element nie mit der Tabulator-Taste angesprungen wird. Um alles Weitere kümmert sich die JavaScript-Funktion `updateProgress()`.

### Kapitel 3 Intelligente Formulare

```
function updateProgress() {
    var req = document.querySelectorAll(":required");
    count = 0;
    for(var i=0; i<req.length; i++) {
        if (req[i].value != '') {
            count++;
        }
    }
    var pb = document.getElementById("formProgress");
    pb.max = req.length;
    pb.value = count;
}
```

Da sich der Fortschrittsbalken nur auf diese Elemente beziehen soll, die unbedingt eingegeben werden müssen, verwenden wir die Funktion `querySelectorAll()` und übergeben ihr die Zeichenkette `:required`. Als Ergebnis erhalten wir eine *NodeList*, die nur Elemente beinhaltet, die das `required`-Attribut gesetzt haben. Anschließend läuft eine Schleife über diese Elemente und überprüft, ob das `value`-Attribut nicht mit einer leeren Zeichenkette übereinstimmt. Trifft diese Bedingung zu (das bedeutet, es wurde schon ein Wert eingegeben), so wird die Zählervariable `count` um einen Wert erhöht. Abschließend wird der Maximalwert für das `progress`-Element auf die Anzahl aller `required`-Felder gesetzt und der Wert (`value`) auf die Zahl der nicht leeren Elemente.

Zum Abschicken des Formulars stehen zwei Schaltflächen zur Verfügung: eine mit der Beschriftung `ZWISCHENSPEICHERN` und eine mit dem Schriftzug `ABSCHICKEN`. Die `Zwischenspeichern`-Funktion wurde bereits in Abschnitt 3.4.5, `Oder doch nicht prüfen? »formnovalidate«`, erklärt, neu ist hier das Attribut `accesskey`.

```
<p><input accesskey=Z type=submit formnovalidate
    value="Zwischenspeichern [Z]" name=save id=save>
<input accesskey=A type=submit name=submit id=submit
    value="Abschicken [A]">
```

Zwar sind Tastaturkürzel nicht neu in HTML5, viel Verwendung fanden sie bisher aber nicht. Ein Problem mit Tastaturkürzeln ist, dass sie auf unterschiedlichen Plattformen mit unterschiedlichen Tastaturkombinationen aktiviert werden und man nie so genau weiß, welche Taste man jetzt zu dem Kürzel drücken muss. Die HTML5-Spezifikation hat auch hierzu einen Vorschlag parat: Der Wert des `accessKeyLabel` soll eine Zeichenkette zurückgeben, die dem korrekten Wert auf der verwendeten Plattform entspricht. Diesen Wert könnte man dann in der Beschriftung der Schaltfläche oder in deren `title`-Attribut

verwenden. Leider war zu dem Zeitpunkt, als dieses Buch geschrieben wurde, kein Browser in der Lage, diese Zeichenkette auszugeben.

So viel zu den neuen Möglichkeiten, die HTML5 für Formulare vorgesehen hat. Für Webentwickler brechen angenehmere Zeiten an, da sie sich nicht mehr mit JavaScript-Bibliotheken für gängige Eingabeelemente wie zum Beispiel Datum und Uhrzeit herumschlagen müssen. Vor allem im Bereich der mobilen Endgeräte, auf denen die Texteingabe meist nicht so angenehm wie am Computer ist, werden die neuen Formular-Funktionen für ein angenehmeres Arbeiten sorgen. Auch die Formular-Überprüfung im Browser wird wesentlich zu einem übersichtlicheren und damit leichter wartbaren Code beitragen. Dabei sollten Sie nicht vergessen, dass die clientseitige Überprüfung kein Sicherheitsgewinn für die Server-Anwendung ist, da es einem Angreifer leicht möglich ist, diese Prüfungen zu umgehen.

Sollten Sie jetzt Lust bekommen haben, das neu erworbene Wissen über Formulare auf Ihrer Webseite auszuprobieren, so können Sie das bereits heute bedenkenlos tun. Die Syntax der neuen Elemente und Attribute ist so aufgebaut, dass auch ältere Browser keine Fehler produzieren. Zwar kommen Benutzer von diesen Browsern nicht in den Genuss der neuen Eingabeelemente und Funktionen, eine Texteingabe ist aber allemal möglich.