Einstieg in JavaScript

Kapitel 1
Einführung

Was ist JavaScript? Was kann ich damit machen und was nicht? Wie baue ich es in meine Internetseite ein? In diesem Kapitel werden erste Fragen geklärt.

JavaScript ist eine objektorientierte Programmiersprache. Sie wurde 1995 erstmals entworfen und wird mithilfe des Standards ECMAScript ständig aktuell gehalten. Seit 2015 gibt es jährlich eine neue Version des Standards. Die Inhalte dieses Buchs basieren auf ECMAScript 2023.

Die Sprache JavaScript wurde für den Einsatz in Internetbrowsern entworfen. Das ist das Thema dieses Einsteigerbuchs. Mittlerweile wird JavaScript aber auch außerhalb von Internetseiten eingesetzt. Trotz Ähnlichkeit in einzelnen Aspekten: JavaScript und die ebenfalls weitverbreitete Programmiersprache Java müssen klar voneinander unterschieden werden.

JavaScript gehört zu den interpretierten Sprachen. Das bedeutet: JavaScript-Programme werden Zeile für Zeile übersetzt und ausgeführt.

Die Sprache bietet viele Elemente, die aus anderen Programmiersprachen bekannt sind, z. B. Schleifen zur schnellen Wiederholung von Programmteilen, Verzweigungen zur unterschiedlichen Behandlung verschiedener Situationen und Funktionen zur Zerlegung eines Programms in übersichtliche Bestandteile. Außerdem steht Ihnen eine Vielfalt von Objekten zur Verfügung. Mithilfe des *Document Object Model* (DOM) haben Sie Zugriff auf alle Elemente Ihrer Internetseiten, sodass Sie sie dynamisch verändern können.

1.1 Was mache ich mit JavaScript?

Die Programme stehen innerhalb von Internetseiten zur Verfügung. Sie enthalten JavaScript-Code gemeinsam mit HTML-Code. JavaScript bietet zusätzliche Möglichkeiten und Hilfen, die HTML nicht hat.

Nach dem Aufruf einer Internetseite mit JavaScript-Code wird dieser Code im Browser ausgeführt und kann die Inhalte der Internetseite dynamisch verändern. Dies geschieht entweder sofort nach dem Laden oder nach dem Eintreten eines Ereignisses, z. B. der Betätigung einer Schaltfläche bei der Benutzung des Programms. JavaScript ermöglicht den Entwurf komplexer Anwendungen mit einer Benutzeroberfläche.

Bei der Entwicklung sollten Sie allerdings darauf achten, dass Ihr Code die Benutzung einer Internetseite nicht einschränkt. Sie sollten z. B. kein JavaScript-Programm schreiben, das das Verlassen einer Internetseite verhindert. Eine solche Internetseite wird kein zweites Mal mehr aufgesucht.

Formulare spielen im Zusammenhang mit JavaScript eine wichtige Rolle:

- Sie dienen der Übermittlung von Daten an einen Webserver. Vor dem Absenden können die Inhalte der Formate durch JavaScript auf Gültigkeit hin überprüft werden. Auf diese Weise wird unnötiger Netzverkehr vermieden.
- Sie ermöglichen eine Interaktion, ähnlich wie man dies von anderen Anwendungen auf dem Rechner gewohnt ist. Es können Eingaben vorgenommen und Verarbeitungen ausgelöst werden. Das Programm liefert anschließend ein Ergebnis zurück.

Zum vereinfachten Entwickeln Ihrer Programme genügt ein Texteditor, der die Markierungen von HTML und die Schlüsselwörter von JavaScript zur Verdeutlichung hervorheben kann. Der Editor Notepad++ ist einer von vielen Editoren, der das beherrscht.

1.2 Was kann JavaScript nicht?

JavaScript kann sich selbst nicht einschalten. Es kann Einzelfälle geben, bei denen JavaScript im Browser ausgeschaltet wurde. Allerdings ist der Anteil an Internetseiten, die dann nicht mehr vollständig genutzt werden können, recht hoch. Wir können aber zumindest erkennen, ob JavaScript ausgeschaltet wurde, und in diesem Fall eine Nachricht übermitteln, siehe Abschnitt 1.11, »Kein JavaScript möglich«.

JavaScript kann (ohne Zusätze) nichts auf dem Webserver speichern. JavaScript-Programme werden im Browser ausgeführt und nicht auf dem Webserver, von dem sie geladen werden. Daher ist es z. B. nicht möglich, Daten auf dem Webserver zu speichern.

JavaScript kann bei der Benutzung nur wenige Daten im Browser speichern. Es kann dort keine Schäden verursachen.

1.3 Browser und mobile Browser

Internetseiten, ob mit oder ohne JavaScript, werden mithilfe von unterschiedlichen Browsern unter verschiedenen Betriebssystemen auf diversen Endgeräten empfangen und bei der Benutzung umgesetzt.

Im Internet werden verschiedene Browserstatistiken geführt. Mit ihrer Hilfe kann festgestellt werden, welche Browser in welcher Häufigkeit verwendet werden. Seit Jahren hat der Browser *Google Chrome* den höchsten Anteil. Daher dient er als wichtigste, aber nicht einzige Referenz für die Beispielprogramme in diesem Buch.

Manche JavaScript-Anweisung kann für bestimmte Browser eventuell anders formuliert werden. Ich empfehle allerdings, immer die Standardform zu benutzen, damit die Anweisung für möglichst viele Browser geeignet ist. Dieser Grundsatz gilt auch für die Beispielprogramme in diesem Buch.

Der Anteil an mobilen Endgeräten mit den dafür zugeschnittenen mobilen Browsern wird immer größer. Mobilgeräte bieten einige zusätzliche Möglichkeiten, z. B. Empfänger bzw. Sensoren für Standortdaten, Lage und Beschleunigung des Mobilgeräts. Die dabei ermittelten Daten können von JavaScript weiterverarbeitet werden, siehe Abschnitt 15.3, »Sensoren«, und folgende.

1.4 ECMAScript

JavaScript basiert auf *ECMAScript*. Sie können in JavaScript objektorientiert programmieren. Dies geschieht klassisch auf der Basis von sogenannten Prototypen und Konstruktorfunktionen. Seit der Einführung des Standards ECMAScript 2015 gibt es eine neue Standardschreibweise, in der mit Klassen, Konstruktoren, Eigenschaften und Methoden gearbeitet wird, wie in anderen objektorientierten Sprachen. Im Buch wird ausschließlich diese Schreibweise genutzt, die auch von allen modernen Browsern umgesetzt wird. Mehr dazu in aller Ausführlichkeit in Kapitel 3, »Eigene Objekte«.

Seit dem Jahr 2015 wird der Standard ECMAScript jährlich aktualisiert. Zurzeit (im Mai 2024) ist der Entwurf für den Standard ECMAScript 2024 in Arbeit. Auf der Internetseite mit der Adresse *https://compat-table.github.io/compat-table/es2016plus* finden Sie eine Übersicht über viele standardisierte Elemente und ihre Umsetzung in den einzelnen Browsern.

Viele Elemente der neueren Standards sind auch für Einsteiger interessant und an der passenden Stelle im Buch zu finden. Ich weise jeweils gesondert auf ihre Einführung hin.

1.5 Aufbau des Buchs

Zunächst eine Anmerkung in eigener Sache: Für die Hilfe bei der Erstellung des Buchs bedanke ich mich beim Team vom Rheinwerk Verlag, besonders bei Anne Scheibe.

Die Themen in diesem Buch stelle ich jeweils mit einer kurzen Beschreibung der Theorie, einem aussagefähigen Screenshot, einem vollständigen, lauffähigen Beispielprogramm und einer ausführlichen praktischen Erläuterung vor. Die Screenshots sind im Browser Google Chrome entstanden, entweder auf einem PC mit Windows 10 oder auf einem Android-Smartphone.

Auf diese Weise haben Sie einen raschen Einstieg in jedes Thema. Sie sind nicht gezwungen, vereinzelte Codezeilen zunächst in einen passenden Kontext zu setzen, um ihre Wirkung zu betrachten. Sie finden alle Beispielprogramme im Downloadmaterial zum Buch, das Sie über *https://www.rheinwerk-verlag.de/einstieg-in-javascript* erreichen.

Im Buch finden Sie Hinweise auf Übungsaufgaben. Die Aufgabenstellungen stehen im Bonuskapitel 1, »Übungen«, im Downloadmaterial zum Buch. Die Übungen geben Ihnen die Möglichkeit, Ihre Kenntnisse zu testen. Eine Lösung zu jeder Übungsaufgabe finden Sie ebenfalls im Downloadmaterial zum Buch.

Die Inhalte des Buchs bauen normalerweise in kleinen, übersichtlichen Schritten aufeinander auf. Dies hat den *Vorteil*, dass die Voraussetzungen zu jedem Thema vorher geklärt sind. Es hat allerdings den *Nachteil*, dass Sie das Buch tatsächlich von vorn nach hinten lesen sollten. Schlagen Sie es dagegen einfach an einer beliebigen Stelle auf, können Sie nicht davon ausgehen, dass an dieser Stelle alle Einzelheiten erklärt werden. Dies ist eventuell in einem früheren Abschnitt geschehen.

Nach der Einleitung in diesem Kapitel 1, »Einführung«, folgen die Grundlagen der Programmierung in Kapitel 2, »Grundlagen der Programmierung«. Hier zeigen sich Ähnlichkeiten mit vielen anderen Programmiersprachen. Objekte spielen in JavaScript eine große Rolle. In Kapitel 3, »Eigene Objekte«, erschaffen Sie eigene Objekte und lernen auf diese Weise ihren Aufbau kennen. In Kapitel 6, »Standardobjekte nutzen«, erläutere ich Ihnen viele vordefinierte Objekte von JavaScript.

Zur Interaktion bei der Benutzung wird mit Ereignissen und ihren Folgen gearbeitet, insbesondere im Zusammenhang mit Formularen, siehe Kapitel 4, »Formulare und Ereignisse«. Die Kenntnis des Aufbaus einer Internetseite nach dem *Document Object*

Model (*DOM*, siehe Kapitel 5, »Das Document Object Model (DOM)«) ermöglicht Ihnen, auf beliebige Stellen im Dokument zuzugreifen und sie zu verändern.

Die *Ajax*-Technologie (siehe Kapitel 7, »Änderungen mit Ajax«) ermöglicht Ihnen u. a. den Austausch einzelner Teile eines Dokuments, ohne eine Seite vollständig neu laden zu müssen. *CSS* (*Cascading Style Sheets*) bieten vielfältige Möglichkeiten der Formatierung und Positionierung von Elementen eines HTML-Dokuments. Diese werden mithilfe von JavaScript dynamisch erweitert, bis hin zur Animation, siehe Kapitel 8, »Gestaltung mit Cascading Style Sheets (CSS)«.

Mithilfe des Standards *SVG* (*Scalable Vector Graphics*) und JavaScript lassen sich dynamische, zweidimensionale Vektorgrafiken erstellen, siehe Kapitel 9, »Zweidimensionale Grafiken und Animationen mit SVG«. Die JavaScript-Bibliothek *Three.js* (siehe Kapitel 10, »Dreidimensionale Grafiken und Animationen mit Three.js«) bietet die Möglichkeit, dreidimensionale Grafiken und Animationen zu entwickeln.

Die weitverbreitete Bibliothek *jQuery* (siehe Kapitel 11, »jQuery«) ermöglicht einen browserunabhängigen, komfortablen Zugriff auf viele Elemente von JavaScript. Die Bibliothek *Onsen UI* (siehe Kapitel 12, »Mobile Apps mit Onsen UI«) dient speziell zur Programmierung mobiler Endgeräte. Mathematische Ausdrücke lassen sich mithilfe von *MathML* und der JavaScript-Bibliothek *MathJax* (siehe Kapitel 13, »Mathematische Ausdrücke mit MathML und MathJax«) in Ihren Dokumenten darstellen.

In Kapitel 14, »Beispielprojekte«, verweise ich auf eine Reihe von größeren, ausführlich kommentierten Beispielprojekten, bei denen das Zusammenspiel vieler Elemente gezeigt wird. Den Zugriff auf Medien und Sensoren sowie die Erstellung von Zeichnungen erläutere ich in Kapitel 15, »Medien, Zeichnungen und Sensoren«.

1.6 Erstes Beispiel mit HTML und CSS

Zum Verständnis der Beispiele dieses Buchs werden nur wenige Kenntnisse in HTML und CSS vorausgesetzt. Die wichtigsten Elemente werden anhand eines ersten Beispielprogramms erläutert.

1.6.1 Ausgabe des Programms

In Abbildung 1.1 und Abbildung 1.2 sehen Sie das Ergebnis des Programms im Browser.

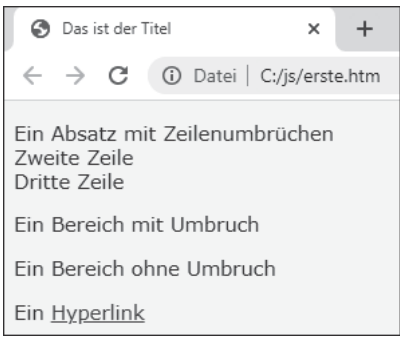

Abbildung 1.1 Erstes HTML-Dokument im Browser, oberer Teil

Abbildung 1.2 Erstes HTML-Dokument im Browser, unterer Teil

1.6.2 HTML-Datei

Das Beispielprogramm enthält einige Grundelemente eines HTML-Dokuments. Es folgt der HTML-Code:

```
<!DOCTYPE html><html lang="de">
<head>
   <meta charset="utf-8">
   <title>Das ist der Titel</title>
   <link rel="stylesheet" href="js5.css">
</head>
<body>
   <p>
      Ein Absatz mit Zeilenumbrüchen<br>
      Zweite Zeile<br>Dritte Zeile
   </p>

   <div>Ein Bereich mit Umbruch</div>
   <p>Ein Bereich <span>ohne</span> Umbruch</p>

   <p>Ein <a href="einbetten.htm">Hyperlink</a></p>
   <p>Ein Bild:<br><img src="im_paradies.jpg" alt="Paradies"></p>
```

```
    <p>Eine Liste:</p>
    <ul>
       <li>Erster Eintrag</li>
       <li>Zweiter Eintrag</li>
    </ul>

    <p>Eine Tabelle:</p>
    <table>
       <tr>
          <td>Zelle A</td>
          <td>Zelle B</td>
       </tr>
       <tr>
          <td>Zelle C</td>
          <td>Zelle D</td>
       </tr>
    </table>
</body>
</html>
```

Listing 1.1 Datei »erste.htm«

Mithilfe von `<!DOCTYPE html>` wird festgelegt, dass es sich um ein HTML-Dokument handelt. Je mehr Sie sich an die einheitlichen Definitionen für HTML-Dokumente halten, desto höher ist die Wahrscheinlichkeit, dass die Seite in allen Browsern fehlerfrei dargestellt wird.

Ein HTML-Dokument besteht aus Markierungen (auch *Tags* genannt) und Text. Die meisten Markierungen bilden einen *Container* (= Behälter), der eine Start-Markierung und eine End-Markierung besitzt. Start-Markierungen können Attribute mit Werten haben. Letztgenannte stehen dabei standardmäßig in doppelten Hochkommata.

Das gesamte Dokument steht im `html`-Container, von der Start-Markierung `<html>` bis zur End-Markierung `</html>`. Die Start-Markierung `html` besitzt das Attribut `lang` mit dem Wert `de`. Damit geben Sie an, dass der Text des Dokuments in deutscher Sprache verfasst ist. Im `html`-Container liegen nacheinander ein `head`-Container mit Informationen über das Dokument und ein `body`-Container mit dem eigentlichen Dokumentinhalt.

Im `head`-Container finden Sie zunächst einen `title`-Container, der den Inhalt für die Titelleiste des Browsers bereitstellt. Außerdem stehen hier Metadaten über das Dokument. Im vorliegenden Beispiel sehen Sie, dass es sich um ein HTML-Dokument handelt,

das den weitverbreiteten Zeichensatz *UTF-8* nutzt, siehe Abschnitt 1.6.3, »Codierung UTF-8«. Er enthält viele Sonderzeichen, z. B. auch die deutschen Umlaute.

Mithilfe der Markierung `link` und den Attributen `rel` und `href` können Sie eine externe CSS-Datei zur Formatierung des Dokuments einbinden. Im vorliegenden Beispiel handelt es sich um die Datei *js5.css*, die in demselben Verzeichnis wie die Datei *erste.htm* liegt. Sie wird in Abschnitt 1.6.4, »Responsives Webdesign«, erläutert.

Absätze stehen in `p`-Containern. Ein einzelner Zeilenumbruch innerhalb eines Absatzes wird mithilfe der Markierung `<br>` gebildet. Bestimmte Bereiche, die eine andere Formatierung erhalten sollen, können Sie sowohl in einen `div`-Container als auch in einen `span`-Container setzen. Vor und nach einem `div`-Container wird zudem ein Umbruch erzeugt.

Ein anklickbarer Hyperlink zu einem anderen Dokument steht in einem `a`-Container mit dem Attribut `href`. Ein Bild kann mithilfe der `img`-Markierung und des Attributs `src` eingebunden werden. Das Attribut `alt` ist für die Validierung erforderlich. Es enthält einen erläuternden Text für den Fall, dass die Bilddatei nicht geladen werden kann.

Eine nicht nummerierte Liste steht in einem `ul`-Container, die einzelnen Listeneinträge stehen in `li`-Containern.

Eine Tabelle wird mithilfe eines `table`-Containers erstellt. Innerhalb der Tabelle gibt es einzelne Zeilen; diese werden jeweils mithilfe eines `tr`-Containers erstellt. Innerhalb einer Zeile wiederum gibt es einzelne Zellen, die jeweils durch einen `td`-Container gebildet werden.

Die Datei *erste.htm* wird mithilfe des Editors Notepad++ erstellt und (bei mir) im Verzeichnis *C:/js* gespeichert. Zur Darstellung einer *htm*-Datei (oder einer *html*-Datei) in Ihrem Standardbrowser öffnen Sie den Windows-Explorer und führen einen Doppelklick auf die *htm*-Datei aus.

Werden weitere HTML-Markierungen genutzt, werden sie an der passenden Stelle erläutert.

1.6.3 Codierung UTF-8

In allen HTML-Dokumenten dieses Buchs wird die Codierung UTF-8 verwendet. *UTF-8* steht abkürzend für das *8-Bit UCS Transformation Format*. *UCS* steht abkürzend für *Universal Character Set*. UTF-8 ist diejenige Codierung für Unicode-Zeichen mit der weitesten Verbreitung.

Es ist wichtig, dass die Codierung, die im `head`-Container angegeben ist, mit der Codierung der Datei übereinstimmt. Sie können, falls noch nicht geschehen, die Codierung einer Datei im Editor Notepad++ wie folgt auf UTF-8 umstellen: Menü Kodierung • Konvertiere zu UTF-8. Anschließend ist in diesem Menü auch die Codierung UTF-8 markiert.

Sie können die Codierung im Editor Notepad++ wie folgt auch automatisch für alle Dateien wählen, die Sie neu erstellen: Menü Einstellungen • Einstellungen • Neues Dokument • Kodierung • UTF-8, Schaltfläche Schliessen.

1.6.4 Responsives Webdesign

In diesem ersten Beispiel wird die externe CSS-Datei *js5.css* zur Formatierung des Dokuments eingebunden. Darin wird mithilfe eines *Media Query* auf vereinfachte Weise ein responsives Webdesign erzeugt. Es werden folgende Ziele erreicht:

- Die Dokumente sind einheitlich formatiert.
- Bei Bedarf kann die Formatierung für alle Dokumente schnell und einheitlich geändert werden.
- Die Beispiele können nicht nur auf einem PC oder einem Laptop genutzt werden, sondern auch auf einem Mobilgerät.

Es folgt der Code in der CSS-Datei:

```
body {font-family:Verdana; font-size:11pt; color:#202020;
      background-color:#f8f8f8;}
td   {font-size:11pt; background-color:#e0e0e0; padding:5px;}

@media only screen and (max-width: 992px)
{
   @media only screen and (orientation:landscape)
   {
      body                 { font-size:20pt; }
      td                   { font-size:20pt; }
      img                  { width:240px; height:180px; }
      input                { font-size:20pt; }
      select               { font-size:20pt; }
      input[type=radio]    { width:30px; height:30px; }
      input[type=checkbox] { width:30px; height:30px; }
      input[type=color]    { width:250px; height:30px; }
```

```
      input[type=range]    { width:250px; height:30px; }
      textarea             { font-size:20pt; height:80px; }
   }
   @media only screen and (orientation:portrait)
   {
      body                 { font-size:32pt; }
      td                   { font-size:36pt; }
      img                  { width:320px; height:240px; }
      input                { font-size:32pt; }
      select               { font-size:32pt; }
      input[type=radio]    { width:45px; height:45px; }
      input[type=checkbox] { width:45px; height:45px; }
      input[type=color]    { width:250px; height:45px; }
      input[type=range]    { width:250px; height:45px; }
      textarea             { font-size:32pt; height:140px; }
   }
}
```

Listing 1.2 Datei »js5.css«

An dieser Stelle folgt nur eine kurze Erläuterung zu CSS. In Kapitel 8, »Gestaltung mit Cascading Style Sheets (CSS)«, sehen Sie mehr zu diesem Thema.

Eine CSS-Angabe kann bezüglich einer HTML-Markierung gelten, auf die sich die Formatierung bezieht. In geschweiften Klammern stehen eine oder mehrere Formatierungen. Diese bestehen wiederum aus einer CSS-Eigenschaft und einem Wert für diese Eigenschaft, getrennt durch einen Doppelpunkt und abgeschlossen durch ein Semikolon.

Im vorliegenden Beispiel wird für die Markierung `body`, also den Inhalt des Dokuments, eine Reihe von Formatierungen vorgenommen:

- Schriftart Verdana, mit `font-family`
- Schriftgröße 11 Punkt, mit `font-size`
- Schriftfarbe `#202020`, mit `color`
- Hintergrundfarbe `#f8f8f8`, mit `background-color`

Farben können mithilfe von RGB-Werten angegeben werden. Nach dem Zeichen `#` folgen jeweils zwei hexadezimale Ziffern für die Farbanteile Rot, Grün und Blau. Die Farbe `#202020` entspricht einem dunklen Grau, die Farbe `#f8f8f8` einem sehr hellen Grau.

Die Schriftgröße innerhalb von Tabellenzellen wird noch einmal gesondert auf 11 Punkt eingestellt. Die Hintergrundfarbe von Tabellenzellen (Markierung `td`) wird mit `#e0e0e0`

auf ein helles Grau gesetzt. Damit sind sie etwas dunkler als der Hintergrund des Dokuments. Mithilfe der Angabe `padding` (deutsch: Polsterung) kann der innere Abstand eines Elements zu seinem umgebenden Element eingestellt werden. Hier wird rund um den Textinhalt einer Tabellenzelle ein Abstand von 5 Pixeln zum Rand der Tabellenzelle gesetzt.

An dieser Stelle enden die CSS-Angaben für die Nutzung auf einem PC oder einem Laptop.

Mithilfe eines Media Query kann auf die Eigenschaften der Geräte reagiert werden, mit denen die Dokumente betrachtet werden. Die CSS-Angabe `@media only screen and (max-width: 992px)` bewirkt, dass sich die CSS-Angaben innerhalb der nachfolgenden geschweiften Klammern:

- nur auf eine Ausgabe auf einem Bildschirm (englisch: *screen*) beziehen, im Unterschied zu einer Ausgabe auf einem Drucker
- nur auf Geräte beziehen, die eine maximale Ausgabebreite von 992 Pixeln besitzen, z. B. Mobilgeräte wie Tablets und Smartphones

Anschließend wird mithilfe der Angabe `orientation` unterschieden, ob sich das Tablet bzw. das Smartphone aktuell im Querformat (Landschaftsmodus) oder im Hochformat (Porträtmodus) befindet.

Die Schriftgröße im Dokument wird im Querformat auf 20 Punkt und im Hochformat auf 32 Punkt vergrößert. In Tabellenzellen findet eine Vergrößerung auf 20 Punkt bzw. 36 Punkt statt.

Viele Bilder in meinen Beispielprogrammen haben zur Vereinfachung eine einheitliche Größe von 160 × 120 Pixeln. Im Querformat werden sie auf 240 × 180 Pixel, im Hochformat auf 320 × 240 Pixel vergrößert.

Die weiteren Elemente begegnen Ihnen erst innerhalb der Formulare ab Kapitel 4, »Formulare und Ereignisse«. Auch ihre Größe wird medienabhängig eingestellt:

- Eingabe allgemein, einzeiliges Textfeld (`input`)
- Auswahl mithilfe eines Menüs (`select`)
- Auswahl mithilfe von Radiobuttons (`input type=radio`)
- Kontrollkästchen (`input type=checkbox`)
- Auswahl einer Farbe (`input type=color`)
- Einstellung einer Zahl in einem Bereich (`input type=range`)
- mehrzeiliges Textfeld (`textarea`)

1.7 Einige Sonderzeichen

Das folgende Programm dient zur Ausgabe einiger Sonderzeichen. Einige davon können Sie auch unmittelbar über die Tastatur eingeben. Alle Sonderzeichen können mithilfe von sogenannten *Entitys* ausgegeben werden. Es folgt das Programm:

```
...
<body>
   <p>Einige Sonderzeichen der Tastatur: > & € @<br>
      dazu auch die Entity: &gt; & &euro; &#64;<br>
      Einige weitere Sonderzeichen: &copy; &reg; &permil;
         &frac14; &frac12; &frac34; &sup2; &sup3; &micro; &pi;<br>
      Das Zeichen &lt; wird nur als Entity validiert</p>
</body></html>
```

Listing 1.3 Datei »sonderzeichen.htm«

Hinweis

In diesem Beispiel und in vielen folgenden Beispielen wird aus Platzgründen der Beginn des Dokuments weggelassen. Er wird nur abgedruckt, falls er zusätzliche Angaben enthält. Zudem wird das Ende des Dokuments kompakter dargestellt.

Die Sonderzeichen in der ersten Zeile können Sie unmittelbar über die Tastatur in Ihr Dokument einfügen: > (größer als), & (Kaufmanns-Und), € (Euro) und @ (at).

Zu vielen Sonderzeichen gibt es sogenannte *Entitys*. Ein Entity beginnt mit dem Zeichen & und endet mit einem Semikolon:

- In der zweiten Zeile sehen Sie die Entitys für >, &, € und @: `>`, `&`, `&euro;` und `&#64;`.
- In der dritten Zeile folgen die Entitys für © (Copyright), ® (Registered Trademark = eingetragenes Warenzeichen), ‰ (Promille), ¼ (ein Viertel), ½ (ein halb), ¾ (drei Viertel), ^{2} (hoch 2) , ^{3} (hoch 3) , µ (mikro) und π (Pi): `&copy;`, `&reg;`, `&permil;`, `&frac14;`, `&frac12;`, `&frac34;`, `&sup2;`, `&sup3;`, `&micro;` und `&pi;`.
- Das Zeichen < (kleiner als) in der vierten Zeile sollten Sie mithilfe des Entitys `<` ausgeben, ansonsten wird das zugehörige Dokument nicht als gültiges HTML-Dokument validiert.

In Abbildung 1.3 sehen Sie das Dokument im Browser.

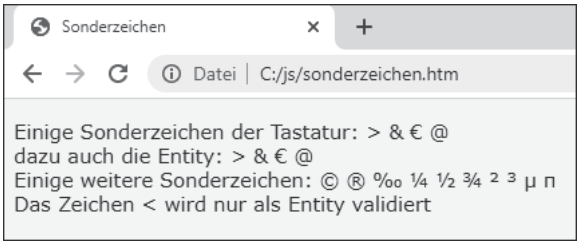

Abbildung 1.3 Einige Sonderzeichen

1.8 JavaScript im Dokument

Nun geht es aber endlich los mit dem ersten JavaScript-Programm. Betrachten Sie zunächst den folgenden Code:

```
...
<body>
   <p>Text in HTML</p>
   <script>
      document.write("<p>Text in JavaScript</p>");
   </script>
   <p>Text in HTML</p>
   <script>
      document.write("<p>JavaScript, doppelte Hochkommata</p>");
      document.write('<p>JavaScript, einfache Hochkommata</p>');

      document.write("<p>Einfache <span style='font-weight:bold;'>"
         + " innerhalb von</span> Doppelten</p>");
      document.write('<p>Doppelte <span style="font-weight:bold;">'
         + ' innerhalb von</span> Einfachen</p>');
   </script>
</body></html>
```

Listing 1.4 Datei »einbetten.htm«

Sie können JavaScript an beliebig vielen Stellen im `head` oder im `body` eines HTML-Dokuments einbetten. Es wird jeweils ein `script`-Container benötigt. Dieser beginnt mit `<script>` und endet mit `</script>`.

Innerhalb des Containers stehen JavaScript-Anweisungen, die nacheinander ausgeführt werden. Sie werden mit einem Semikolon abgeschlossen.

Der Aufruf `document.write()` sorgt für die Ausgabe einer Zeichenkette. Zwischen dem Objekt `document` und der Methode `write()` steht der *Punkt-Operator*. Damit wird die Methode `write()` für das Objekt `document` aufgerufen. Den Begriff *Objekt* erläutere ich in Kapitel 3, »Eigene Objekte«, in aller Ausführlichkeit.

Zeichenketten stehen standardmäßig in doppelten Hochkommata. Innerhalb der Zeichenketten können sowohl Texte als auch HTML-Markierungen stehen. Sie können Zeichenketten aber auch in einfache Hochkommata kleiden.

Dasselbe gilt für den Wert von Attributen, wie hier beim Wert des Attributs `style`. Sie werden in HTML standardmäßig in doppelten Hochkommata notiert. Sie können Werte von Attributen aber auch in einfache Hochkommata kleiden.

Damit es bei der Ausgabe von Attributwerten, die innerhalb einer Zeichenkette für JavaScript stehen, nicht zu Fehlern kommt, müssen unterschiedliche Hochkommata kombiniert werden. Das sind entweder einfache Hochkommata innerhalb von doppelten Hochkommata oder umgekehrt.

Ich habe die CSS-Eigenschaft `font-weight` mit dem Wert `bold` genutzt, um einen Teil des Textes in Fettschrift zu setzen.

Einige Hinweise

- Beachten Sie beim Programmieren die richtige Schreibweise der Anweisungen. Die Browser verzeihen, anders als in HTML, in JavaScript keine Fehler.
- JavaScript unterscheidet zwischen Groß- und Kleinschreibung. Mit der Anweisung `document.Write(...)` werden Sie keinen Erfolg haben, da es die Methode `Write()` mit großem Anfangsbuchstaben `W` nicht gibt.
- Sie können auch mehrere Anweisungen in eine Zeile schreiben. Hauptsache, es steht ein Semikolon am Ende jeder Anweisung.

In Abbildung 1.4 sehen Sie verschiedene Teile des Dokuments, die zum Teil aus HTML und zum Teil aus JavaScript stammen.

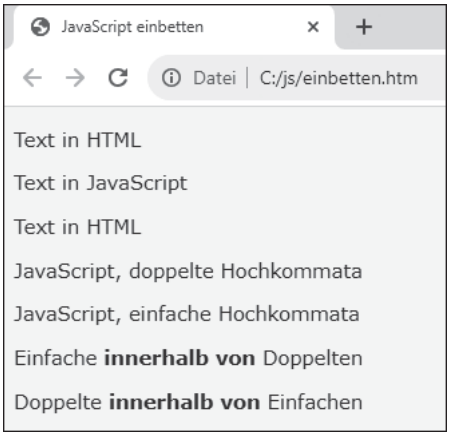

Abbildung 1.4 JavaScript innerhalb einer Datei

1.9 JavaScript aus externer Datei

Sie können Programmteile, die Sie in mehreren JavaScript-Programmen nutzen möchten, in einer externen Datei speichern. Auf den Code einer solchen externen Datei können Sie leicht zugreifen, indem Sie die Datei in Ihr Programm einbinden. Es folgt ein Beispiel:

```
...
<body>
   <script src="externe_datei.js"></script>
   <script>
      document.write("<p>Das kommt aus extern.htm</p>");
   </script>
</body></html>
```

Listing 1.5 Datei »extern.htm«

Der erste `script`-Container ist leer. Allerdings wird das Attribut `src` mit dem Wert `externe_datei.js` notiert. Damit wird der Code aus der betreffenden Datei in die Datei *extern.htm* eingebunden. In der Datei *externe_datei.js* steht lediglich der folgende Code:

```
document.write("<p>Das kommt aus externe_datei.js</p>");
```

Listing 1.6 Datei »externe_datei.js«

In Abbildung 1.5 sehen Sie die beiden Absätze, die jeweils mithilfe der Methode `document.write()` aus dem zusammengefügten Programm erzeugt werden.

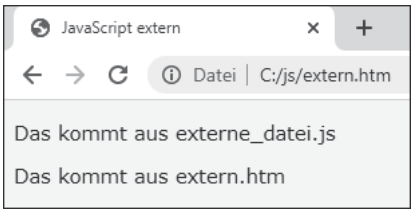

Abbildung 1.5 Zusätzliches JavaScript aus externer Datei

Beachten Sie, dass in der externen Datei kein `script`-Container steht. Der Name dieser Datei kann eine beliebige Endung haben. Als Konvention hat sich die Endung *js* eingebürgert.

Auf die genannte Weise werden die Bibliothek jQuery (siehe Kapitel 11, »jQuery«) und andere große JavaScript-Bibliotheken mit ihren vielen nützlichen Funktionen in Anwendungen eingebunden.

1.10 Kommentare

Kommentare dienen zur Beschreibung der einzelnen Teile Ihrer Programme. Sie erleichtern Ihnen und anderen das Verständnis eines Programms. Betrachten wir ein Beispiel:

```
...
<body>
   <!-- Das ist ein Kommentar
        im HTML-Bereich -->
   <p>Ein Absatz aus dem HTML-Bereich</p>
   <script>
      /* Das ist ein Kommentar über mehrere Zeilen
         im JavaScript-Bereich */
      document.write("<p>Ein Absatz aus dem JS-Bereich</p>");
      // Ein kurzer Kommentar, nur bis zum Zeilenende
   </script>
</body></html>
```

Listing 1.7 Datei »kommentar.htm«

Im Beispiel sehen Sie drei verschiedene Arten von Kommentaren:

- Ein Kommentar im HTML-Bereich kann sich über eine oder über mehrere Zeilen erstrecken. Er steht zwischen den Zeichenfolgen `<!--` und `-->`.
- Im JavaScript-Bereich wird ein Kommentar, der über eine oder mehrere Zeilen geht, zwischen den Zeichenfolgen `/*` und `*/` notiert.
- Möchten Sie einen kurzen Kommentar im JavaScript-Bereich notieren, beispielsweise hinter einer Anweisung, so eignet sich die Zeichenfolge `//`. Ein solcher einzeiliger Kommentar geht nur bis zum Ende der jeweiligen Zeile.

Der Inhalt der Kommentare wird nicht im Browser dargestellt, siehe Abbildung 1.6. Allerdings kann jeder bei Bedarf den Quelltext einer Seite in seinem Browser ansehen und damit auch die Kommentare.

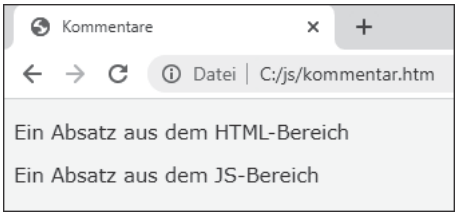

Abbildung 1.6 Kommentare sind nicht sichtbar.

Hinweis

Häufig möchten Sie ein Programm, das von Ihnen oder von jemand anderem stammt, nach längerer Zeit noch einmal ansehen oder erweitern. Dann werden Sie für jede Zeile Kommentar dankbar sein, die Sie darin vorfinden. Aus den gleichen Gründen ist es auch sehr zu empfehlen, übersichtliche, leicht lesbare Programme zu schreiben.

1.11 Kein JavaScript möglich

Wie bereits in Abschnitt 1.2, »Was kann JavaScript nicht?«, erwähnt: Es kann Einzelfälle geben, bei denen JavaScript im Browser ausgeschaltet wurde. Da JavaScript sich selbst nicht einschalten kann: Was können wir machen?

Wir können erkennen, ob es eingeschaltet ist oder nicht. Ist es nicht eingeschaltet, können wir entweder eine einfache Version der Seite in reinem HTML anbieten oder einen Hinweis geben, dass die Nutzung der betreffenden Seite das Einschalten von JavaScript voraussetzt.

Ein Beispiel:

```
...
<body>
   <script>
      document.write("<p>Hier läuft JavaScript</p>");
   </script>
   <noscript>
      <p>Hier läuft JavaScript nicht<br>
         Bitte schalten Sie es ein</p>
   </noscript>
</body></html>
```

Listing 1.8 Datei »kein_script.htm«

Innerhalb des `noscript`-Containers können Sie Text und HTML-Markierungen für den Fall notieren, dass JavaScript ausgeschaltet ist.

Bei eingeschaltetem JavaScript werden nur die Anweisungen aus dem `script`-Container ausgeführt. Die Seite sieht dann aus wie in Abbildung 1.7.

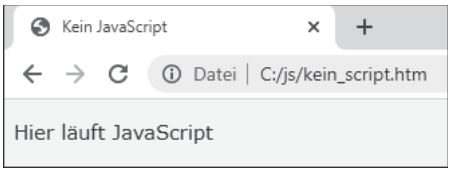

Abbildung 1.7 JavaScript ist eingeschaltet.

Sie können JavaScript einmal zu Testzwecken ausschalten. Die notwendige Vorgehensweise wird dazu nachfolgend am Beispiel des Browsers Google Chrome erläutert. Rufen Sie über die drei Punkte oben rechts das Menü auf. Wählen Sie den Menüpunkt EINSTELLUNGEN. Geben Sie im Suchfenster den Begriff »JavaScript« ein. In den Suchergebnissen finden Sie die WEBSITE-EINSTELLUNGEN. Dort finden Sie den Eintrag JAVASCRIPT, standardmäßig mit dem Eintrag ZUGELASSEN. Wählen Sie den Eintrag über den Pfeil rechts aus, und stellen Sie den Schalter auf BLOCKIERT.

Anschließend sehen Sie beim Aufruf einer Datei, die JavaScript enthält, ganz rechts in der Adresszeile des Browsers ein Symbol mit der Information, dass JavaScript auf dieser Internetseite blockiert wurde. Klicken Sie auf das Symbol, öffnet sich ein Dialogfeld. Hier

haben Sie die Möglichkeit, JavaScript *für diese Seite* einzuschalten. Über die Schaltfläche VERWALTEN gelangen Sie auch unmittelbar zu dem oben genannten Schalter zum Ein- und Ausschalten von JavaScript, und zwar allgemein oder für einzelne Seiten.

Bei ausgeschaltetem JavaScript sieht die Seite *kein_script.htm* aus wie in Abbildung 1.8.

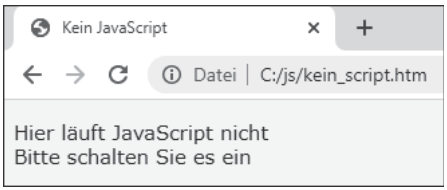

Abbildung 1.8 JavaScript ist ausgeschaltet.

Kapitel 4
Formulare und Ereignisse

Sie lernen die Kontrolle von Formularen und die Behandlung von Ereignissen kennen.

Formulare ermöglichen Interaktionen, ähnlich wie man dies von anderen Anwendungen auf dem Rechner gewohnt ist. Es können Eingaben vorgenommen und Verarbeitungen ausgelöst werden. Das Programm liefert anschließend ein Ergebnis.

Zudem dienen Formulare der Übermittlung von Daten an einen Webserver. Vor dem Absenden können ihre Inhalte durch JavaScript auf Gültigkeit hin überprüft werden. Auf diese Weise wird unnötiger Netzverkehr vermieden.

JavaScript ermöglicht Ihnen, Programme zu schreiben, die auf Ereignisse (englisch: *events*) im Browser reagieren können. Auf diese Weise kann mit Ihrem Programm interagiert werden, und zwar über die einfachen Rückgabewerte der Funktionen `prompt()` und `confirm()` hinaus.

Bei diesen Ereignissen kann es sich um den Klick auf eine Schaltfläche, die Auswahl eines Eintrags aus einer Liste, eine bestimmte Aktion mit der Maus, das Absenden eines Formulars und vieles mehr handeln.

Soll Ihr Programm auf ein Ereignis reagieren, entwickeln Sie JavaScript-Programmcode mit Anweisungen, die im Fall des Ereignisses ausgeführt werden sollen. Ein *Eventhandler* dient zur Behandlung des Ereignisses. Sie stellen damit eine Verbindung zwischen dem Teil des Dokuments, bei dem das Ereignis ausgelöst wird, dem Ereignis selbst und dem JavaScript-Programmcode her.

4.1 Erstes Formular und erstes Ereignis

In diesem Abschnitt lernen Sie einige typische Elemente kennen, die zur Behandlung eines Ereignisses benötigt werden.

Im nachfolgenden Beispiel sehen Sie ein Formular mit dem Eingabefeld EINGABE und der Schaltfläche (englisch: *button*) KLICKEN. Nach der Eingabe eines Texts im Eingabe-

feld wird die Schaltfläche betätigt, siehe Abbildung 4.1. Anschließend erscheint ein Dialogfeld mit einer Information, siehe Abbildung 4.2.

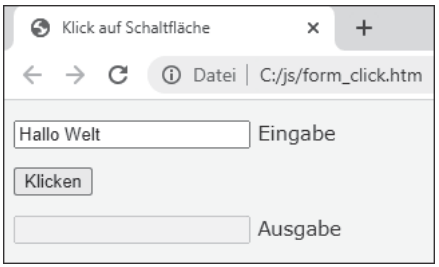

Abbildung 4.1 Formular mit Eingabefeld und Schaltfläche

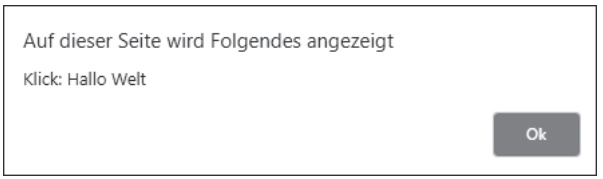

Abbildung 4.2 Reaktion auf Ereignis

Eingabefelder können auch zur nachträglichen Ausgabe von Informationen innerhalb von Dokumenten genutzt werden, die bereits vollständig abgebildet wurden. Im Beispiel erscheint im Feld AUSGABE dieselbe Information wie im Dialogfeld, siehe Abbildung 4.3. Mithilfe des Attributs `disabled` kann dafür gesorgt werden, dass in diesem Feld keine Eingaben gemacht werden können.

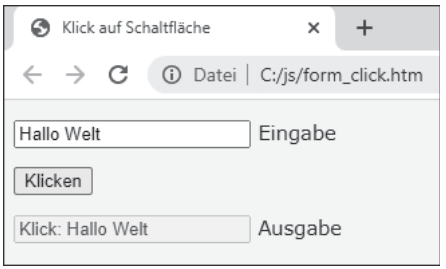

Abbildung 4.3 Feld für Ausgabe

Es folgt das Programm:

```
... <head> ...
   <script>
      function geklickt()
```

```
      {
         const eingabefeld = document.getElementById("idEingabe");
         const eingabe = eingabefeld.value;
         alert("Klick: " + eingabe);
         const ausgabefeld = document.getElementById("idAusgabe");
         ausgabefeld.value = "Klick: " + eingabe;
      }
   </script>
</head>
<body>
   <form>
      <p><input id="idEingabe"> Eingabe</p>
      <p><input id="idButton" type="button" value="Klicken"></p>
      <p><input id="idAusgabe" disabled> Ausgabe</p>
   </form>
   <script>
      const bu = document.getElementById("idButton");
      bu.addEventListener("click", geklickt);
   </script>
</body></html>
```

Listing 4.1 Datei »form_click.htm«

Die Elemente eines Formulars stehen in einem `form`-Container. Ein einfaches Element des Typs `input` entspricht einem einzeiligen Eingabefeld. Wird für das Attribut `type` eines `input`-Elements der Wert `button` angegeben, wird eine Schaltfläche dargestellt. Der Wert des Attributs `value` entspricht dem Text auf der Schaltfläche.

Der Wert des Attributs `id` dient als eindeutige Identifikation (kurz: ID) des Elements im Dokument. Diese ID wird für den Zugriff auf das Element und die Verbindung zum JavaScript-Programmcode benötigt.

Im unteren `script`-Container wird die Methode `getElementById()` des `document`-Objekts aufgerufen. Sie erwartet als Parameter die ID eines Elements und liefert einen Verweis auf ein Objekt zurück, das dieses Element repräsentiert. Dieser Verweis wird hier in der Variablen `bu` gespeichert. Die Variable `bu` verweist damit auf die Schaltfläche.

Ein *Eventlistener* registriert ein Ereignis und reagiert darauf. Die Methode `addEventListener()` des `element`-Objekts verbindet ein Element des Dokuments mit einem bestimmten Ereignis und mit einem Verweis auf eine benannte oder eine anonyme Funktion. Im vorliegenden Fall bedeutet das: Der Klick auf die Schaltfläche `bu` löst das Ereignis `click` aus und damit einen Aufruf der benannten Funktion `geklickt()`.

In der Funktion `geklickt()` verweist die Variable `eingabefeld` auf das Eingabefeld. Die Eigenschaft `value` des Eingabefelds enthält den eingetragenen Text. Dieser Text wird im Dialogfeld ausgegeben.

Die Variable `ausgabefeld` verweist auf das Eingabefeld, das zur Ausgabe genutzt wird. Der Eigenschaft `value` kann auch ein Wert zugewiesen werden. Damit ändert sich der Inhalt des Eingabefelds.

Hinweis

Zum Thema »Formulare« finden Sie die Übungsaufgabe »u_formular« im Downloadmaterial zum Buch.

Hinweis

Seit der Anfangszeit von JavaScript werden unterschiedliche Techniken für die Verbindung zwischen dem Element eines Dokuments und dem Programmcode genutzt. Moderne Browser kennen einheitlich die Methoden `getElementById()` und `addEventListener()` und ihre vielfältigen Einsatzmöglichkeiten.

4.2 Senden und Zurücksetzen

Bezüglich eines Formulars können die Ereignisse *Senden* (englisch: *to submit*) und *Zurücksetzen* (englisch: *to reset*) auftreten. Daten können mithilfe von Formularen an einen Webserver gesendet werden. Beim Zurücksetzen werden wieder die Startwerte des Formulars eingestellt.

Im nachfolgenden Beispiel wird auf diese beiden Ereignisse reagiert. Zudem wird der Inhalt des Formulars an eine Datei mit einem PHP-Programm gesendet, die auf einem Webserver liegt und die Daten weiterverarbeitet. Im vorliegenden Beispiel werden die übertragenen Daten nur auf dem Bildschirm ausgegeben.

Bei PHP handelt es sich um eine weitverbreitete Programmiersprache für den Einsatz auf Webservern. Zum Verständnis des Beispiels müssen Sie aber kein PHP erlernen. Es geht nur darum, den vollständigen Ablauf zu verdeutlichen.

4.2.1 Der Ablauf beim Senden

In Abbildung 4.4 sehen Sie den Aufruf der Datei *form_senden.htm*. Sie wird, wie in den bisherigen Beispielen, aus dem Verzeichnis der JavaScript-Programme aufgerufen, bei mir *C:\js*.

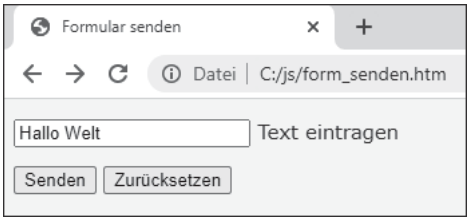

Abbildung 4.4 Aufruf des Formulars aus dem Verzeichnis

In Abbildung 4.5 wird die Reaktion von JavaScript auf das Absenden gezeigt.

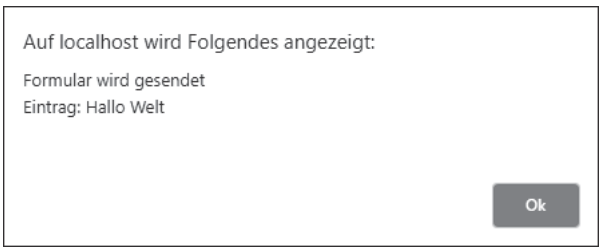

Abbildung 4.5 Reaktion von JavaScript auf das Absenden

Da der gesamte Vorgang nicht auf einem Webserver stattfindet, kann die PHP-Datei nicht ordnungsgemäß aufgerufen werden. Es wird nur ihr Code abgebildet, siehe Abbildung 4.6. Eine Abhilfe zu diesem Problem folgt im nächsten Abschnitt 4.2.2, »Webserver als Alternative«.

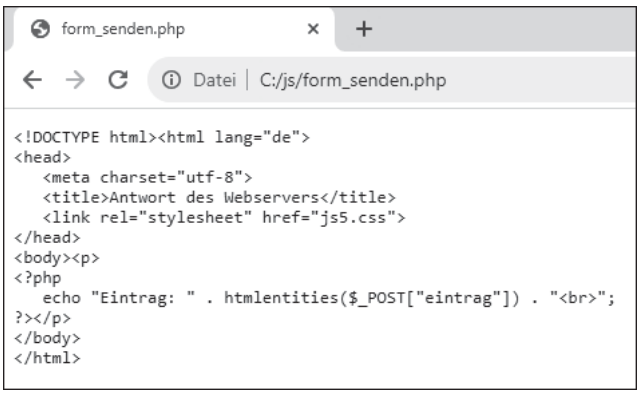

Abbildung 4.6 PHP-Code der Antwort

Wenn Sie das Formular zurücksetzen, statt es abzusenden, erfolgt die Reaktion in Abbildung 4.7.

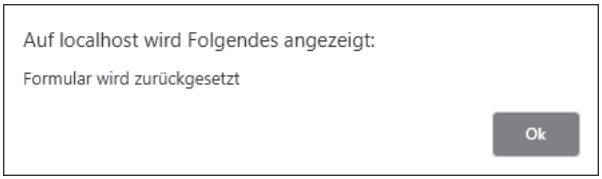

Abbildung 4.7 Reaktion auf das Zurücksetzen des Formulars

4.2.2 Webserver als Alternative

Möchten Sie den Ablauf der gesamten Übertragung auf einem Webserver verfolgen, gibt es zwei Alternativen:

Alternative 1: Sie können einen lokalen Webserver auf Ihrem Rechner nutzen und das Formular über diesen Webserver aufrufen. Dazu installieren Sie das frei verfügbare Programmpaket XAMPP und speichern die beiden beteiligten Dateien *form_senden.htm* und *form_senden.php* in einem Verzeichnis dieses Webservers. Das kann z. B. das Verzeichnis *js5* unterhalb des Basisverzeichnisses sein. Nach dem Aufruf der Datei *form_senden.htm* und dem Absenden des Formulars sieht die Antwort des PHP-Programms dann aus wie in Abbildung 4.8. Die Installation und die Nutzung von XAMPP werden in Abschnitt A.1, »Installation des Pakets XAMPP«, beschrieben.

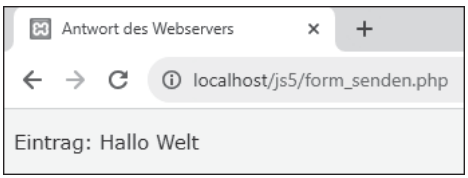

Abbildung 4.8 Antwort über lokalen Webserver

Alternative 2: Haben Sie die Möglichkeit, Daten auf einem Internetserver zu speichern? Dann laden Sie die beiden beteiligten Dateien *form_senden.htm* und *form_senden.php* in ein Verzeichnis dieses Internetservers hoch. Ich habe die Dateien im Verzeichnis *js5* auf meiner Domain *theisweb.de* gespeichert.

Die Adresse des Verzeichnisses ist *https://theisweb.de/js5*. Hier finden Sie Hyperlinks zu allen Programmen dieses Buchs, nach ihrem Auftreten in den Kapiteln geordnet. Die weiteren Beispiele dieses Kapitels werden einheitlich über meine Domain aufgerufen, auch wenn das nicht bei allen Programmen notwendig ist.

Nach dem Aufruf der Datei *form_senden.htm* und dem Absenden des Formulars sieht die Antwort des PHP-Programms aus wie in Abbildung 4.9.

Abbildung 4.9 Antwort über Internetserver

Sollten Sie keine der beiden Alternativen verwenden und die Programme mit den Formularen nur aus dem Verzeichnis heraus aufrufen, ist das kein Problem. Mit dem JavaScript-Code können Sie in jedem Fall arbeiten.

4.2.3 Code zum Senden

Es folgt der Code der Datei mit dem gesendeten Formular:

```
... <head> ...
   <script>
      function senden()
      {
         const tx = document.getElementById("idText");
         const eintrag = tx.value;
         alert("Formular wird gesendet\nEintrag: " + eintrag);
      }

      function zuruecksetzen()
      {
         alert("Formular wird zurückgesetzt");
      }
   </script>
</head>
<body>
   <form id="idForm" method="post" action="form_senden.php">
      <p><input id="idText" name="eintrag"> Text eintragen</p>
      <p><input type="submit"> <input type="reset"></p>
   </form>
   <script>
      const fo = document.getElementById("idForm");
      fo.addEventListener("submit", senden);
      fo.addEventListener("reset", zuruecksetzen);
```

```
   </script>
</body></html>
```

Listing 4.2 Datei »form_senden.htm«

Das Formular besitzt die eindeutige ID `idForm`. Der Wert `post` für das Attribut `method` dient zum Festlegen der sicheren Übertragungsmethode *Post*, die in allen Beispielen gewählt wird. Der Wert des Attributs `action` verweist auf das antwortende PHP-Programm. Es wird vorausgesetzt, dass es in demselben Verzeichnis liegt.

Das Eingabefeld besitzt die eindeutige ID `idText`, die für den Zugriff auf den Inhalt des Felds mit JavaScript benötigt wird. Zudem besitzt das Eingabefeld das Attribut `name`, hier mit dem Wert `eintrag`, das für den Zugriff auf den Inhalt des Felds mit PHP benötigt wird.

Die Werte `submit` bzw. `reset` für das Attribut `type` der beiden `input`-Elemente kennzeichnen diese als Schaltflächen zum Senden bzw. Zurücksetzen des Formulars.

Die Variable `fo` verweist auf das Formular. Mithilfe der Methode `addEventListener()` werden die beiden Ereignisse `submit` und `reset` mit den benannten Funktionen `senden()` bzw. `zuruecksetzen()` verbunden.

4.2.4 Code zum Empfangen

Wie bereits erwähnt: Sie müssen kein PHP erlernen. Zur Information und zur Verdeutlichung folgt an dieser Stelle aber *ausnahmsweise* der Code der empfangenden PHP-Datei, wie Sie ihn bereits in Abbildung 4.6 gesehen haben:

```
<!DOCTYPE html><html lang="de">
<head>
   <meta charset="utf-8">
   <title>Antwort des Webservers</title>
   <link rel="stylesheet" href="js5.css">
</head>
<body><p>
<?php
   echo "Eintrag: " . htmlentities($_POST["eintrag"]) . "<br>";
?></p>
</body>
</html>
```

Listing 4.3 Datei »form_senden.php«

Es handelt sich um ein Standard-HTML-Dokument. Der PHP-Code ist in einem Container zwischen `<?php` und `?>` eingebettet. Das Sprachelement `echo` dient zur Ausgabe auf dem Bildschirm. Die Punkte verbinden die einzelnen Textteile miteinander, wie das Pluszeichen in JavaScript.

Diejenigen Elemente, die im Formular das Attribut `name` besitzen, werden mit ihren Attributwerten automatisch zu Elementen des PHP-Felds `$_POST`. Also: Aus `name="eintrag"` wird `$_POST["eintrag"]`.

Die Funktion `htmlentities()` dient zur Absicherung vor der Wirkung von Schadcode, den ein böswilliger Nutzer mithilfe des Eingabefelds zum Webserver übertragen könnte.

Zu fast allen Formularbeispielen dieses Buchs gibt es zur Verdeutlichung des Ablaufs eine antwortende PHP-Datei.

Haben Sie weitergehendes Interesse an PHP, verweise ich auf mein Buch: *https://www.rheinwerk-verlag.de/einstieg-in-php-und-mysql*.

4.3 Pflichtfelder und Kontrolle

Sie können die Eingaben in einem Formular bereits mit HTML auf einfache Weise validieren, d. h. auf Gültigkeit hin prüfen. Eine weitergehende Validierung der eingegebenen oder ausgewählten Inhalte eines Formulars ist mit JavaScript möglich.

Im folgenden Beispiel sollen ein Nachname und ein Passwort in zwei Pflichtfelder eingegeben werden, siehe Abbildung 4.10. Zusätzlich kann eine Bemerkung in einem mehrzeiligen Eingabefeld eingetragen werden. Beim Betätigen der Schaltfläche SENDEN passiert Folgendes:

- Ist eines der beiden Pflichtfelder leer, wird eine Fehlermeldung im Formular angezeigt (siehe Abbildung 4.11), und es wird nicht gesendet.
- Wird ein Passwort mit weniger als drei Zeichen oder mehr als acht Zeichen eingegeben, wird eine Fehlermeldung von JavaScript angezeigt (siehe Abbildung 4.12), und das Formular wird nicht gesendet.
- Sind beide Pflichtfelder gefüllt und hat das Passwort die richtige Länge, wird das Formular gesendet.

Sobald Sie mehr Kenntnisse im Aufbau und in der Behandlung von Zeichenketten haben, sind weitergehende Prüfungen möglich. In Abschnitt 6.2.5, »Prüfen eines Passworts«, finden Sie ein größeres Beispiel.

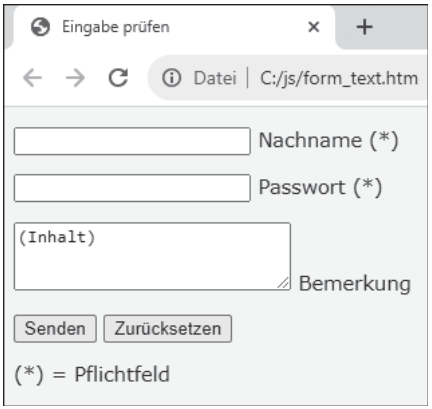

Abbildung 4.10 Pflichtfelder und mehrzeiliges Eingabefeld

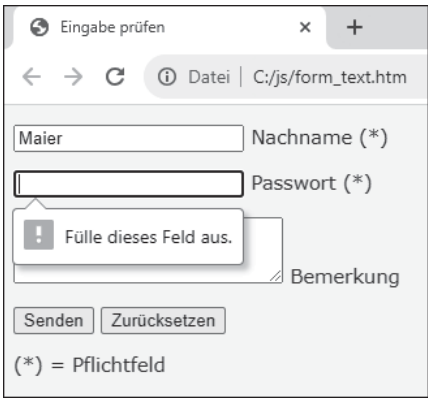

Abbildung 4.11 Fehlermeldung von HTML

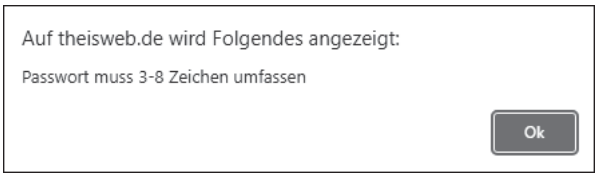

Abbildung 4.12 Fehlermeldung von JavaScript

Es folgt der Code der Datei mit dem Formular:

```
... <head> ...
   <script>
      function senden(e)
```

```
      {
         const pw = document.getElementById("idPasswort").value;
         if(pw.length < 3 || pw.length > 8)
         {
            alert("Passwort muss 3-8 Zeichen umfassen");
            e.preventDefault();
            return false;
         }

         alert(document.getElementById("idNachname").value
            + "\n" + document.getElementById("idPasswort").value
            + "\n" + document.getElementById("idBemerkung").value);
      }
   </script>
</head>
<body>
   <form id="idForm" method="post" action="form_text.php">
      <p><input id="idNachname" name="nachname"
            required="required"> Nachname (*)</p>
      <p><input id="idPasswort" name="passwort"
            type="password" required="required"> Passwort (*)</p>
      <p><textarea id="idBemerkung" rows="3" cols="25"
            name="bemerkung">(Inhalt)</textarea> Bemerkung</p>
      <p><input type="submit"> <input type="reset"></p>
      <p>(*) = Pflichtfeld</p>
   </form>
   <script>
      document.getElementById("idForm").addEventListener
         ("submit", function(e) { return senden(e);});
   </script>
</body></html>
```

Listing 4.4 Datei »form_text.htm«

Der Aufbau des Formulars:

Der Wert `password` für das Attribut `type` eines `input`-Elements macht dieses Element zu einem Eingabefeld für ein Passwort. Nur die Anzahl der eingegebenen Zeichen ist zu erkennen. Die Zeichen selbst sind nicht zu lesen.

Ein mehrzeiliges Eingabefeld wird mithilfe eines `textarea`-Containers dargestellt. Die Größe richtet sich zu Beginn nach den Werten der Attribute `rows` (deutsch: *Zeilen*) und

`cols` (Abkürzung für columns, deutsch: *Spalten*). Meist lässt sie sich im Browser verändern. Der Eintrag steht in JavaScript über die Eigenschaft `value` zur Verfügung, wie bei einzeiligen Eingabefeldern.

Die beiden Eingabefelder für den Nachnamen und das Passwort besitzen das Attribut `required` mit dem Wert `required` (deutsch: *erforderlich*). Die Felder dürfen daher nicht leer gelassen werden. Ist keines der beiden Eingabefelder leer, wird das Ereignis `submit` ausgelöst.

Der Eventhandler:

Für den Verweis auf das Formular wird unmittelbar die Methode `addEventListener()` aufgerufen. Diese verkürzte Schreibweise wird auch in den meisten nachfolgenden Beispielen genutzt.

Die Übergabe eines Parameters beim Verbinden eines Ereignisses mit einer Funktion ist nur mithilfe einer anonymen Funktion möglich. Im vorliegenden Beispiel handelt es sich bei dem Parameter um einen Verweis auf das `event`-Objekt (deutsch: *Ereignisobjekt*), das bei einem Ereignis automatisch mit weiteren Informationen zur Verfügung steht. Gemäß Konvention erhält der Verweis den Namen `e`.

In der anonymen Funktion wird die benannte Funktion `senden()` aufgerufen. Dabei wird der Parameter der anonymen Funktion wiederum übergeben. Die Funktion `senden()` liefert einen Wahrheitswert an die anonyme Funktion zurück. Ist dieser `false`, wird das Formular nicht abgesendet.

Der Ablauf in der Funktion `senden()`:

In der Funktion `senden()` wird das eingegebene Passwort ermittelt, indem die Eigenschaft `value` unmittelbar für den Verweis auf das Eingabefeld aufgerufen wird. Auch hier handelt es sich um eine verkürzende Schreibweise, die ebenso in den nachfolgenden Beispielen genutzt wird.

Die Eigenschaft `length` einer Zeichenkette enthält die Anzahl der Zeichen einer Zeichenkette. Mehr zu Zeichenketten finden Sie in Abschnitt 6.2, »Zeichenketten verarbeiten«. Ist das Passwort zu kurz oder zu lang, liefert die Funktion `senden()` den Wert `false` zurück, und es erfolgt eine Fehlermeldung.

Zudem wird die Methode `preventDefault()` des `event`-Objekts aufgerufen, damit die Standardaktion für das Element, hier das Absenden des Formulars, nicht durchgeführt wird.

Hat das Passwort die richtige Länge, werden die drei Eingaben angezeigt. Zudem liefert die Funktion `senden()` den Wert `true`, und das Formular wird gesendet.

4.4 Radiobuttons und Checkboxen

Eine Gruppe von Radiobuttons (deutsch: *Optionsschaltflächen*) dient zur Auswahl zwischen verschiedenen Möglichkeiten. Mithilfe einer Checkbox (deutsch: *Kontrollkästchen*) kann eine zusätzliche Auswahl getroffen werden. Mithilfe von JavaScript stellen Sie fest, welche Auswahl getroffen wurde.

Im nachfolgenden Beispiel kann eines von drei verschiedenen Ländern gewählt werden. Zusätzlich können noch zwei weitere Länder ausgewählt werden, siehe Abbildung 4.13.

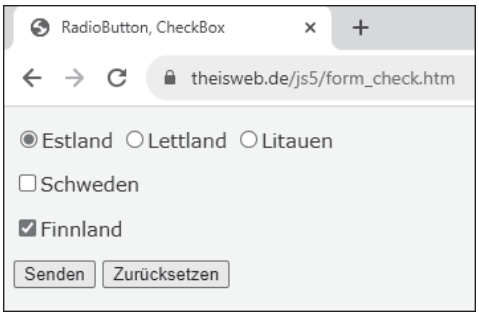

Abbildung 4.13 Radiobuttons und Checkboxen

Beim Absenden des Formulars werden die Werte der verschiedenen Auswahlelemente angezeigt, siehe Abbildung 4.14. Diese müssen nicht mit den angezeigten Texten übereinstimmen.

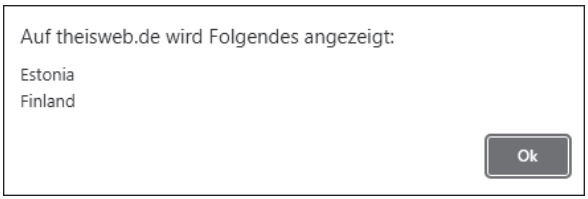

Abbildung 4.14 Werte der Auswahlelemente

Es folgt der Code der Datei mit dem Formular:

```
... <head> ...
   <script>
      function senden()
      {
         let tx;

         if(r1.checked) tx = r1.value;
```

```
            else if(r2.checked) tx = r2.value;
            else if(r3.checked) tx = r3.value;

            if(c1.checked) tx += "\n" + c1.value;
            if(c2.checked) tx += "\n" + c2.value;

            alert(tx);
         }
      </script>
   </head>
   <body>
      <form id="idForm" method="post" action="form_check.php">
         <p><input id="idEstland" name="baltikum" type="radio"
               value="Estonia" checked="checked">Estland
            <input id="idLettland" name="baltikum" type="radio"
               value="Latvia">Lettland
            <input id="idLitauen" name="baltikum" type="radio"
               value="Lithuania">Litauen</p>
         <p><input id="idSchweden" name="schweden" type="checkbox"
               value="Sweden">Schweden</p>
         <p><input id="idFinnland" name="finnland" type="checkbox"
               value="Finland" checked="checked">Finnland</p>
         <p><input type="submit"> <input type="reset"></p>
      </form>
      <script>
         const r1 = document.getElementById("idEstland");
         const r2 = document.getElementById("idLettland");
         const r3 = document.getElementById("idLitauen");
         const c1 = document.getElementById("idSchweden");
         const c2 = document.getElementById("idFinnland");
         document.getElementById("idForm")
            .addEventListener("submit", senden);
      </script>
   </body></html>
```

Listing 4.5 Datei »form_check.htm«

Der Aufbau des Formulars:

Die Attribute `radio` und `checkbox` kennzeichnen die `input`-Elemente als Radiobuttons bzw. Checkboxen. Das Attribut `value` enthält den Wert, der für das Element übermittelt wird.

Alle Radiobuttons mit demselben Wert für das Attribut `name` gehören zusammen. Wird eine Auswahl innerhalb der Gruppe getroffen, wird die bisherige Auswahl zurückgenommen.

Einer der Radiobuttons innerhalb einer Gruppe sollte bereits mithilfe des Attributs `checked` und des Werts `checked` markiert sein, damit die Gruppe immer einen Wert besitzt.

Die Verweise auf die einzelnen Elemente werden gespeichert.

Der Ablauf in der Funktion `senden()`:

Die Gruppe von Radiobuttons wird mithilfe einer mehrfachen Verzweigung geprüft. Die Checkboxen werden jeweils mithilfe einer einfachen Verzweigung geprüft. Hat die Eigenschaft `checked` den Wert `true`, ist das betreffende Element markiert. Die Eigenschaft `value` enthält auch hier den Wert des Elements.

Hinweis

Zum Thema »Radiobuttons« finden Sie die Übungsaufgabe »u_radio« im Downloadmaterial zum Buch.

4.5 Auswahlmenüs

Ein einfaches Auswahlmenü dient ebenfalls zur Auswahl zwischen verschiedenen Möglichkeiten. In einem mehrfachen Auswahlmenü können mithilfe der Tasten ⇧ und Strg mehrere Optionen gleichzeitig ausgewählt werden. Mithilfe von JavaScript können Sie feststellen, welche Auswahl getroffen wurde.

Im nachfolgenden Beispiel sehen Sie ein einfaches Auswahlmenü und ein mehrfaches Auswahlmenü. Beide Menüs enthalten jeweils drei Länder, siehe Abbildung 4.15.

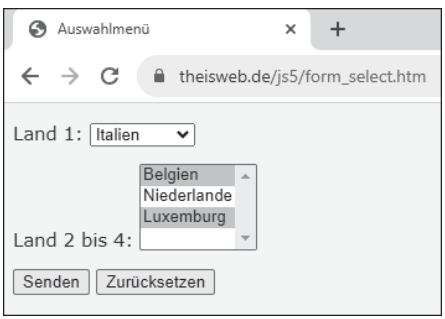

Abbildung 4.15 Einfaches und mehrfaches Auswahlmenü

Beim Absenden des Formulars werden die Werte der verschiedenen Auswahlelemente angezeigt, siehe Abbildung 4.16. Diese müssen nicht mit den angezeigten Texten übereinstimmen.

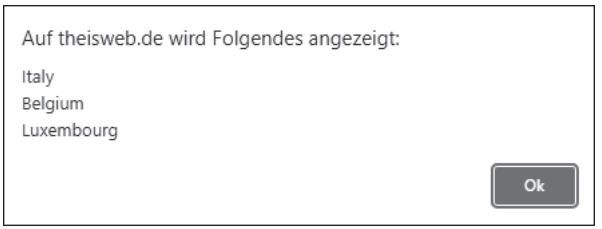

Abbildung 4.16 Werte der Auswahlelemente

Es folgt der Code der Datei mit dem Formular:

```
... <head> ...
   <script>
      function senden()
      {
         let tx = document.getElementById("idEine").value;

         const opt = document.getElementById("idMehrere").options;
         for(let i=0; i<opt.length; i++)
            if(opt[i].selected)
               tx += "\n" + opt[i].value;

         alert(tx);
      }
   </script>
</head>
<body>
   <form id="idForm" method="post" action="form_select.php">
      <p>Land 1: <select id="idEine" name="eine">
            <option value="Italy" selected="selected">Italien</option>
            <option value="Spain">Spanien</option>
            <option value="Romania">Rumänien</option>
         </select></p>
      <p>Land 2 bis 4:
         <select id="idMehrere" name="mehrere[]" multiple="multiple">
            <option value="Belgium" selected="selected">Belgien</option>
            <option value="Netherlands">Niederlande</option>
```

```
            <option value="Luxembourg" selected="selected">Luxemburg</option>
         </select></p>
      <p><input type="submit"> <input type="reset"></p>
   </form>
   <script>
      document.getElementById("idForm").addEventListener("submit", senden);
   </script>
</body></html>
```

Listing 4.6 Datei »form_select.htm«

Der Aufbau des Formulars:

Ein Auswahlmenü steht in einem `select`-Container. Mithilfe des Attributs `multiple` und des Werts `multiple` wird es zu einem mehrfachen Auswahlmenü. Die einzelnen Optionen stehen in `option`-Containern. Die Werte der einzelnen Optionen werden mithilfe des Attributs `value` festgelegt.

Ähnlich wie bei den Radiobuttons sollte bei einem einfachen Auswahlmenü eine der Optionen mithilfe des Attributs `selected` und des Werts `selected` ausgewählt sein, damit das Menü immer einen Wert hat.

Bei einem mehrfachen Auswahlmenü empfiehlt es sich, bei dem Wert für das Attribut `name` rechteckige Klammern anzuhängen. Damit können die einzelnen Elemente bei der Auswertung in dem PHP-Programm wie Feldelemente behandelt werden.

Der Ablauf in der Funktion `senden()`:

Der Wert eines einfachen Auswahlmenüs lässt sich leicht ermitteln, da nur eine Option ausgewählt werden kann. Die Eigenschaft `value` des Menüs enthält den gewünschten Wert.

Beim mehrfachen Auswahlmenü ist die Ermittlung aufwendiger, da mehrere Werte ausgewählt werden können.

Bei der Eigenschaft `options` eines Menüs handelt es sich um ein Feld mit Verweisen auf die einzelnen Optionen. Die Eigenschaft `length` entspricht der Größe des Felds, also der Anzahl der Optionen. Auf eine einzelne Option wird mithilfe von rechteckigen Klammern zugegriffen.

Jede Option besitzt die Eigenschaft `selected`. Ist die Option ausgewählt, hat die Eigenschaft den Wert `true`, ansonsten `false`. Die Eigenschaft `value` einer Option enthält ihren Wert.

Hinweis

Zum Thema »Auswahlmenü« finden Sie die Übungsaufgabe »u_select« im Downloadmaterial zum Buch.

4.6 Weitere Formular-Ereignisse

Sie haben bereits die Ereignisse `click` (auf Schaltfläche) sowie `submit` und `reset` (eines Formulars) kennengelernt.

Im nachfolgenden Programm kommen weitere Ereignisse hinzu, die innerhalb eines Formulars stattfinden können:

- `click`: Klicken auf einem Radiobutton bzw. einer Checkbox
- `change`: Ändern des Inhalts eines Eingabefelds bzw. der Auswahl in einem Auswahlmenü
- `keyup`: Loslassen einer Taste in einem Eingabefeld
- `focus`: Element wird betreten, d. h., es erhält den Eingabefokus, und der Eingabecursor steht in dem Element
- `blur`: Element wird verlassen, d. h., es verliert den Eingabefokus, und der Eingabecursor steht nicht mehr in dem Element

Die Reaktionen auf die verschiedenen Ereignisse werden in einem schreibgeschützten Eingabefeld angezeigt. Als Beispiel sehen Sie in Abbildung 4.17 die Ausgabe nach dem Klicken auf eine Checkbox, um sie zu markieren.

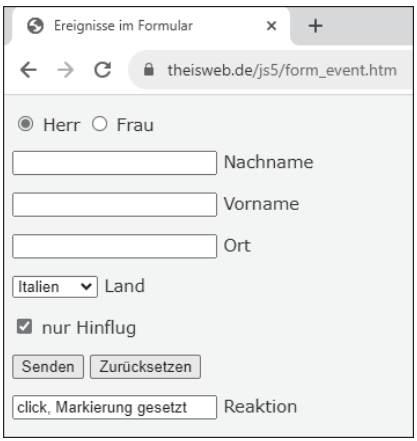

Abbildung 4.17 Reaktion auf Markieren der Checkbox

Es folgt der Code der Datei mit dem Formular:

```
... <head> ...
   <script>
      function wert(e)
      {
         const re = document.getElementById("idReaktion");
         re.value = e.type + ", ";

         const id = e.target.id;
         if(id == "idReise")
            if(document.getElementById(id).checked)
               re.value += "Markierung gesetzt";
            else
               re.value += "Markierung gelöscht";
         else
            re.value += document.getElementById(id).value;
      }
   </script>
</head>
<body id="idBody">
   <form method="post" action="form_event.php">
      <p><input id="idHerr" name="anrede" type="radio"
            value="Herr" checked="checked"> Herr
         <input id="idFrau" name="anrede" type="radio"
            value="Frau"> Frau</p>
      <p><input id="idNachname" name="nachname"> Nachname</p>
      <p><input id="idVorname" name="vorname"> Vorname</p>
      <p><input id="idOrt" name="ort"> Ort</p>
      <p><select id="idLand" name="land">
            <option value="Italien" selected="selected">Italien</option>
            <option value="Spanien">Spanien</option>
            <option value="Portugal">Portugal</option>
         </select> Land</p>
      <p><input id="idReise" name="reise" type="checkbox"
         value="Hinflug"> nur Hinflug</p>
      <p><input type="submit"> <input type="reset"></p>
      <p><input id="idReaktion"
         readonly="readonly"> Reaktion</p>
   </form>
```

```
   <script>
      document.getElementById("idHerr").addEventListener
         ("click",  function(e) { wert(e); });
      document.getElementById("idFrau").addEventListener
         ("click",  function(e) { wert(e); });
      document.getElementById("idNachname").addEventListener
         ("change", function(e) { wert(e); });
      document.getElementById("idVorname").addEventListener
         ("keyup",  function(e) { wert(e); });
      document.getElementById("idOrt").addEventListener
         ("focus",  function(e) { wert(e); });
      document.getElementById("idOrt").addEventListener
         ("blur",  function(e) { wert(e); });
      document.getElementById("idLand").addEventListener
         ("change", function(e) { wert(e); });
      document.getElementById("idReise").addEventListener
         ("click",  function(e) { wert(e); });
   </script>
</body></html>
```

Listing 4.7 Datei »form_event.htm«

Der Aufbau des Formulars und der Eventhandler:

Das Attribut `readonly` mit dem Wert `readonly` sorgt dafür, dass kein Eintrag in dem Eingabefeld möglich ist. Es dient nur zur Ausgabe von Informationen.

Bei der Auswahl eines der beiden Radiobuttons wird das Ereignis `click` ausgelöst. Das Ereignis ist mit einer anonymen Funktion verbunden, die die benannte Funktion `wert()` aufruft. Dabei wird der Parameter `e` weitergereicht, der auf ein Ereignisobjekt verweist. Auf dieselbe Weise werden auch noch weitere Ereignisse mit der Funktion `wert()` verbunden.

In einem Eingabefeld findet das Ereignis `change` statt, nachdem sich der Inhalt des Eingabefelds geändert hat und sobald ein anderes Element ausgewählt wird bzw. außerhalb des Eingabefelds geklickt wird.

Das Ereignis `keyup` in einem Eingabefeld wird unmittelbar nach dem Loslassen einer Taste ausgelöst. Dieses Ereignis tritt nach dem Eingeben bzw. nach dem Löschen eines Zeichens auf.

In einem Auswahlmenü findet das Ereignis `change` statt, sobald die Auswahl gewechselt wird. Wird eine Checkbox markiert bzw. die Markierung einer Checkbox entfernt, wird das Ereignis `click` ausgelöst.

Die Ereignisse `focus` und `blur` finden in einem Eingabefeld statt, sobald das Eingabefeld betreten oder anschließend wieder verlassen wird.

Der Ablauf in der Funktion `wert()`:

Die Eigenschaft `target` des Ereignisobjekts verweist auf das Element, bei dem das Ereignis ausgelöst wurde. Die Eigenschaft `type` enthält den Namen des Ereignisses. Die Untereigenschaft `id` der Eigenschaft `target` entspricht der eindeutigen ID des Elements. Auf diese Weise können Sie eine Funktion für mehrere Elemente bzw. Ereignisse verwenden.

Bei der Checkbox wird die Eigenschaft `checked` geprüft, um festzustellen, ob beim Ereignis `click` die Markierung gesetzt oder entfernt wurde. Für alle anderen Elemente wird der aktuelle Wert desjenigen Elements ausgegeben, bei dem das Ereignis ausgelöst wurde.

Testen Sie einmal die möglichen Ereignisse, und beachten Sie den Zeitpunkt und den Inhalt der Reaktion.

4.7 Maus-Ereignisse

Unabhängig von Formularen gibt es noch weitere Ereignisse, die mithilfe der Maus ausgelöst werden können. Im nachfolgenden Programm werden folgende behandelt:

- `click`: Mausklick auf einem Element
- `mousemove`: Bewegen der Maus über einem Element
- `mousedown`: Herunterdrücken einer Maustaste über einem Element
- `mouseup`: Loslassen einer Maustaste über einem Element
- `mouseover`: Betreten eines Elements
- `mouseout`: Verlassen eines Elements

Zur Verdeutlichung finden diese Ereignisse auf Bildern statt. In einem schreibgeschützten Eingabefeld werden Informationen zu den Ereignissen ausgegeben, siehe Abbildung 4.18.

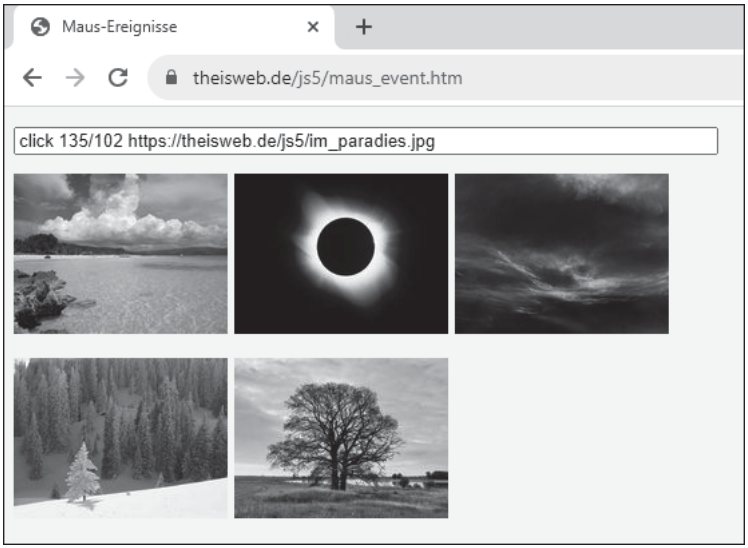

Abbildung 4.18 Maus-Ereignisse

Es folgt der Code der Datei:

```
... <head> ...
   <script>
      function maus(e)
      {
         document.getElementById("idReaktion").value = e.type + " " + e.offsetX
            + "/" + e.offsetY + " " + document.getElementById(e.target.id).src;
      }
   </script>
</head>
<body>
   <form>
      <p><input size="70" id="idReaktion" readonly="readonly"></p>
   </form>
   <p><img src="im_paradies.jpg" id="idParadies" alt="Paradies">
      <img src="im_sofi.jpg"     id="idSofi"     alt="Sofi">
      <img src="im_welle.jpg"    id="idWelle"    alt="Welle"></p>
   <p><img src="im_winter.jpg"   id="idWinter"   alt="Winter">
      <img src="im_baum.jpg"     id="idBaum"     alt="Baum"></p>
```

```
  <script>
    document.getElementById("idParadies").addEventListener
      ("click", function(e) { maus(e); });
    document.getElementById("idSofi").addEventListener
      ("mousemove", function(e) { maus(e); });
    document.getElementById("idWelle").addEventListener
      ("mousedown", function(e) { maus(e); });
    document.getElementById("idWelle").addEventListener
      ("mouseup", function(e) { maus(e); });

    document.getElementById("idWinter").addEventListener
      ("mouseover", function(e) { maus(e); });
    document.getElementById("idWinter").addEventListener
      ("mouseout", function(e) { maus(e); });

    const baum = document.getElementById("idBaum");
    baum.addEventListener("mouseover",
      function() { baum.src = "im_paradies.jpg"; });
    baum.addEventListener("mouseout",
      function() { baum.src = "im_baum.jpg"; });
  </script>
</body></html>
```

Listing 4.8 Datei »maus_event.htm«

Der Aufbau der Eventhandler:

Ein Klick auf das erste Bild löst das Ereignis `click` aus. Es ist mit einer anonymen Funktion verbunden, die die benannte Funktion `maus()` aufruft. Dabei wird der Parameter `e` weitergereicht, der auf ein Ereignisobjekt verweist. Auf dieselbe Weise werden auch noch weitere Ereignisse mit der Funktion `maus()` verbunden.

Ein `img`-Element besitzt die Eigenschaft `src`, die den Namen der Bilddatei enthält, gegebenenfalls mit Pfad. Beim letzten Bild wird ein Bildwechsel durchgeführt, indem bei den Ereignissen `mouseover` bzw. `mouseout` eine anonyme Funktion aufgerufen wird. Innerhalb der Funktion erhält die Eigenschaft `src` jeweils einen neuen Wert.

In der Funktion `maus()` werden drei Informationen ausgegeben:

- der Name des Maus-Ereignisses
- die relativen Koordinaten (`offsetX` und `offsetY`) derjenigen Stelle innerhalb des Elements, an der das Maus-Ereignis stattgefunden hat
- der Name der Bilddatei, die in dem betreffenden `img`-Element dargestellt wird

Testen Sie einmal die möglichen Ereignisse, und beachten Sie den Ort des Ereignisses, die zugehörige Information und den Zeitpunkt des Erscheinens der Information.

Hinweis

Zum Thema »Maus-Ereignisse« finden Sie die Übungsaufgabe »u_maus« im Downloadmaterial zum Buch.

4.8 Wechsel des Dokuments

Im Standardfall werden Hyperlinks genutzt, um zu einem anderen Dokument im Browser zu wechseln. Mithilfe des vordefinierten `location`-Objekts und der passenden Ereignisse geht das auch anders.

Im nachfolgenden Programm (siehe Abbildung 4.19) geschieht das:

- nach einem Klick auf eine Schaltfläche
- nach einem Klick auf ein Bild
- durch den Wechsel der Auswahl in einem Auswahlmenü

Das vordefinierte `location`-Objekt enthält Informationen über den aktuellen *URL* (*Uniform Resource Locator*), also über die Ressource, die im Browser angezeigt wird, z. B. ein HTML-Dokument.

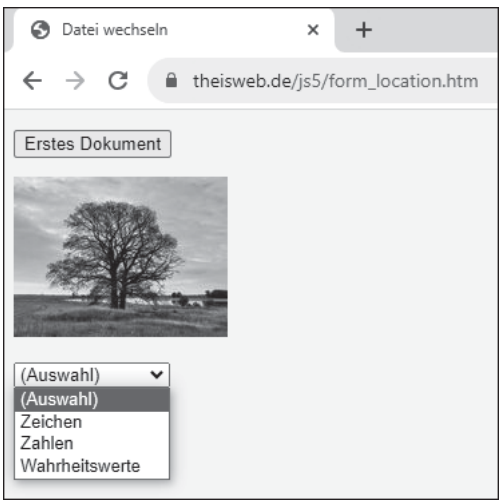

Abbildung 4.19 Wechsel des Dokuments

Es folgt der Code der Datei:

```
...
<body>
   <form>
      <p><input id="idButton" type="button"
         value="Erstes Dokument"></p>
      <p><img src="im_baum.jpg" id="idBaum" alt="Baum"></p>
      <p><select id="idAuswahl">
            <option value="(Auswahl)" selected="selected">(Auswahl)</option>
            <option value="zeichen.htm">Zeichen</option>
            <option value="zahlen.htm">Zahlen</option>
            <option value="bool.htm">Wahrheitswerte</option>
         </select> </p>
   </form>
   <script>
      document.getElementById("idButton").addEventListener
         ("click", function() { location.href = "erste.htm"; } );
      document.getElementById("idBaum").addEventListener("click",
         function() { location.href = "sonderzeichen.htm"; } );

      const auswahl = document.getElementById("idAuswahl");
      auswahl.addEventListener("change", function() {
         if(auswahl.value != "(Auswahl)")
            location.href = auswahl.value; } );
   </script>
</body></html>
```

Listing 4.9 Datei »form_location.htm«

Das Ereignis `click` auf der Schaltfläche wird mit einer anonymen Funktion verbunden. Darin wird der Eigenschaft `href` des `location`-Objekts ein Dateiname als neuer Wert zugewiesen. Das bewirkt den Wechsel zu dem gewünschten Dokument. Dasselbe geschieht bei einem Klick auf das Bild.

Das Ereignis `change` beim Wechsel der Auswahl in dem Auswahlmenü wird ebenfalls mit einer anonymen Funktion verbunden. Darin wird geprüft, ob der Wert der aktuell gewählten Option, also der Wert des gesamten Auswahlmenüs, dem Wert der ersten Option entspricht. Diese erste Option dient als Titel des Auswahlmenüs und soll nicht zu einem Wechsel des Dokuments führen. Handelt es sich um eine der anderen Optionen, wird zu dem gewünschten Dokument gewechselt.

4.9 Weitere Typen und Eigenschaften

Es gibt weitere Typen und Eigenschaften von Eingabefeldern, die besonders seit der Version 5 von HTML zur Verfügung gestellt werden. Sie bieten vielfältige und leicht bedienbare Möglichkeiten.

Die verschiedenen Typen können eine Validierung enthalten, also eine Prüfung des Elements beim Absenden des Formulars. Das dient dazu, die Übermittlung fehlerhafter Daten zu vermeiden. Diese Fähigkeit kann Formularprüfungen durch JavaScript ergänzen, eventuell sogar ersetzen. Prüfen Sie vor der Erstellung eines JavaScript-Programms, ob die gewünschte Prüfung nicht bereits mithilfe von HTML in den modernen Browsern durchgeführt werden kann.

Setzt ein einzelner Browser einen bestimmten Typ von Eingabefeld nicht um, wird stattdessen ein Standardeingabefeld angezeigt. So wird in jedem Fall eine Eingabe ermöglicht. Besonders die mobilen Browser nutzen viele Eigenschaften und unterstützen Sie durch das Einblenden einer Tastatur mit jeweils passenden Elementen.

4.9.1 Texteingaben, Suchfelder und Farben

Im nachfolgenden Programm werden Eingabefelder für verschiedene Textarten, Suchfelder mit Listen und Elemente zur Auswahl von Farben vorgestellt. Außerdem wird die Verwendung besonderer Eigenschaften erläutert.

Das gesamte Dokument sehen Sie in Abbildung 4.20.

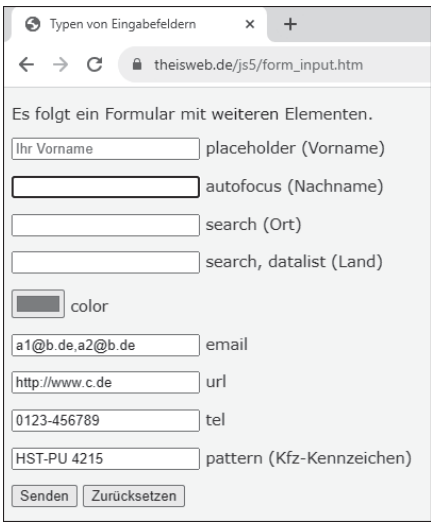

Abbildung 4.20 Weitere Typen und Eigenschaften

Es folgt der Code der Datei:

```
... <head> ...
   <script>
      function senden()
      {
         alert(document.getElementById("idVorname").value
            + "\n" + document.getElementById("idNachname").value
            + "\n" + document.getElementById("idPlz").value
            + " " + document.getElementById("idOrt").value
            + "\n" + document.getElementById("idLand").value
            + "\n" + document.getElementById("idFarbe").value
            + "\n" + document.getElementById("idAdressen").value
            + "\n" + document.getElementById("idWeb").value
            + "\n" + document.getElementById("idTelefon").value
            + "\n" + document.getElementById("idKfz").value);
      }
   </script>
</head>
<body>
   <p>Es folgt ein Formular mit <mark>weiteren</mark> Elementen.</p>
   <form id="idForm" method="post" action="form_input.php">
      <p><input id="idVorname" name="vorname"
         placeholder="Ihr Vorname"> placeholder (Vorname)</p>
      <p><input id="idNachname" name="nachname" autofocus>
         autofocus (Nachname)</p>
      <input id="idPlz" name="plz" type="hidden" value="53484">

      <p><input id="idOrt" name="ort" type="search"> search (Ort)</p>
      <p><input id="idLand" name="land" type="search"
         list="idListe"> search, datalist (Land)</p>
      <datalist id="idListe">
         <option value="Deutschland">
         <option value="Frankreich">
         <option value="Italien">
      </datalist>

      <p><input id="idFarbe" name="farbe" type="color" value="#ff0000"> color</p>
      <p><input id="idAdressen" name="adressen" type="email"
         multiple="multiple" value="a1@b.de, a2@b.de"> email</p>
      <p><input id="idWeb" name="web" type="url" value="http://www.c.de"> url</p>
```

```
      <p><input id="idTelefon" name="telefon" type="tel"
         value="0123-456789"> tel</p>
      <p><input id="idKfz" name="kfz" value="HST-PU 4215"
         pattern="^[A-Z]{1,3}-[A-Z]{1,2} [1-9][0-9]{0,3}$">
         pattern (Kfz-Kennzeichen)</p>
      <p><input type="submit"> <input type="reset"></p>
   </form>

   <script>
      document.getElementById("idForm").addEventListener("submit", senden);
   </script>
</body></html>
```

Listing 4.10 Datei »form_input.htm«

Ein `mark`-Container dient zur optischen Hervorhebung von Text, z. B. mit einer unregelmäßigen gelben Markierung ähnlich der eines Textmarkers. Er kann zur Kennzeichnung der wichtigen Elemente eines Formulars genutzt werden.

Im Element für den Vornamen erscheint mithilfe des Attributs `placeholder` ein Platzhalter in hellgrauer Farbe. Er dient zur Erläuterung eines Eingabefelds. Wird mit einer Eingabe im Eingabefeld begonnen, verschwindet der Platzhalter.

Nach dem Aufruf des Programms erscheint der Eingabecursor dank des Attributs `autofocus` automatisch im Element für den Nachnamen. Dort kann unmittelbar mit der Eingabe begonnen werden. Sollte in einem solchen Element bereits ein Text stehen, wird dieser vollständig markiert.

Bereits seit Langem gibt es die Möglichkeit, ein Eingabefeld mithilfe des Werts `hidden` (deutsch: *versteckt*) für das Attribut `type` zu verstecken. Auf diese Weise können zusätzliche Daten an den Webserver übermittelt werden, die bei der Benutzung nicht sichtbar sein sollen.

Der Wert `search` für das Attribut `type` kennzeichnet ein Eingabefeld, das als typisches Suchfeld zu erkennen ist. Es kann auch ein Symbol zum Löschen des Suchbegriffs enthalten.

Ein Suchfeld kann das zusätzliche Attribut `list` haben. Sein Wert entspricht der ID einer Liste mit vorgegebenen Einträgen, die bei der Bedienung des Suchfelds ausgewählt werden können, siehe auch Abbildung 4.21. Sobald Zeichen im Suchfeld eingetragen werden, erscheinen nur noch diejenigen Einträge, in denen die Zeichen vorkommen. Zusätzlich können weitere Suchbegriffe eingetragen werden. Die Liste wird mithilfe eines

`datalist`-Containers gebildet. Jeder Eintrag stellt den Wert des Attributs `value` einer `option`-Markierung dar.

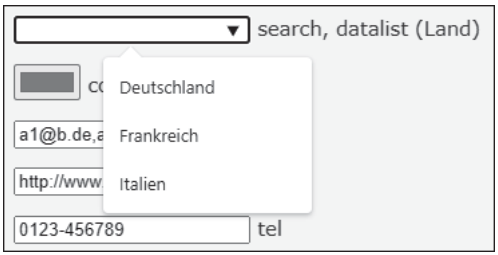

Abbildung 4.21 Suchfeld mit Liste

Der Wert `color` für das Attribut `type` kennzeichnet ein Eingabefeld zur Einstellung einer Farbe. Nach Auswahl des Eingabefelds öffnet sich ein Dialogfeld mit Einstellmöglichkeiten, siehe auch Abbildung 4.22.

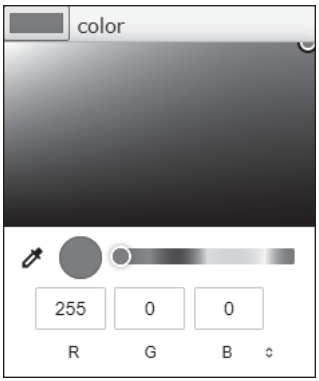

Abbildung 4.22 Farbe einstellen

In einem Eingabefeld mit dem Wert `email` für das Attribut `type` kann eine E-Mail-Adresse eingegeben werden. Das Feld kann leer sein. Ist es nicht leer, muss es die Mindestbestandteile einer E-Mail-Adresse enthalten. Ansonsten wird das Formular nicht abgesendet. Wird das Attribut `multiple` mit dem Wert `multiple` hinzugefügt, können mehrere Adressen eingetragen werden, durch Kommata voneinander getrennt.

Ähnlich sieht es bei einem Eingabefeld mit dem Wert `url` für das Attribut `type` aus. Das Feld kann leer sein. Ist es nicht leer, muss es die Mindestbestandteile eines URL enthalten. Ansonsten wird das Formular nicht abgesendet. Allerdings können nicht mehrere URLs eingetragen werden.

Das Element zur Eingabe einer Telefonnummer besitzt den Wert `tel` für das Attribut `type`. Es kann die Möglichkeit bieten, auf vorhandene Telefonlisten zuzugreifen.

Das Attribut `pattern` (deutsch: *Muster*) dient zur Validierung einer Eingabe mithilfe von regulären Ausdrücken. Erfolgt ein Eintrag, der nicht zum Muster passt, ist das Absenden des Formulars nicht möglich.

Im vorliegenden Fall soll ein deutsches Kfz-Kennzeichen eingetragen werden. Dieses besteht aus ein bis drei großen Buchstaben, einem Bindestrich, ein bis zwei weiteren großen Buchstaben, einem Leerzeichen und einer bis vier Ziffern. Die erste Ziffer darf keine 0 sein. Mehr zu regulären Ausdrücken finden Sie im Bonuskapitel 2, »Reguläre Ausdrücke«, im Downloadmaterial zum Buch.

Nach dem Absenden wird die Funktion `senden()` aufgerufen. Alle Werte werden mithilfe der Eigenschaft `value` angezeigt.

4.9.2 Elemente für Zahlen

In diesem Abschnitt zeige ich einige Elemente, die zur sicheren Eingabe und zur anschaulichen Darstellung von Zahlenwerten geeignet sind, siehe Abbildung 4.23.

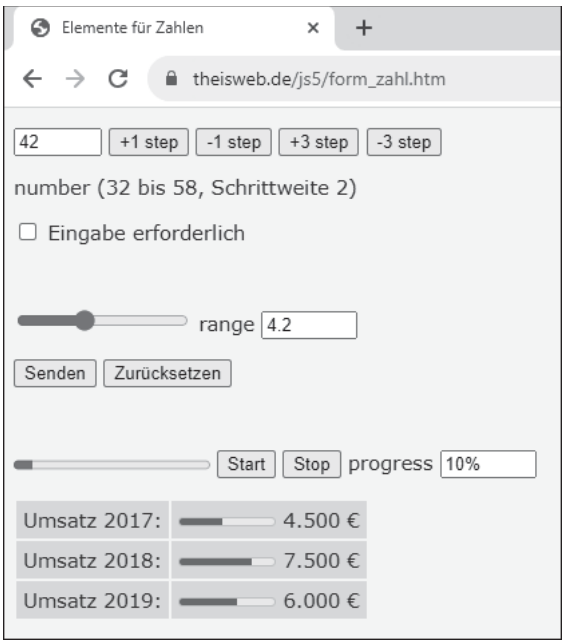

Abbildung 4.23 Elemente für Zahlen

Es folgt zunächst der Inhalt des Formulars im mittleren Teil der Datei:

```
...
<body>
   <form id="idForm" method="post" action="form_zahl.php">
      <p><input id="idZahl" name="zahl" type="number"
            min="32" max="58" step="2" value="42">
         <input id="idPlus1" type="button" value="+1 step">
         <input id="idMinus1" type="button" value="-1 step">
         <input id="idPlus3" type="button" value="+3 step">
         <input id="idMinus3" type="button" value="-3 step"></p>
      <p>number (32 bis 58, Schrittweite 2)</p>
      <p><input id="idRequired" type="checkbox"> Eingabe erforderlich</p><br>
      <p><input id="idBereich" name="bereich" type="range" min="3.2"
            max="5.8" step="0.2" value="4.2"> range <input id="idBWert"
            readonly="readonly" size="5" value="4.2"></p>
      <p><input type="submit"> <input type="reset"></p><br>

      <p><progress id="idFortschritt" max="100" value="10"></progress>
         <input id="idStart" type="button" value="Start">
         <input id="idStop" type="button" value="Stop"> progress <input
            id="idFWert" readonly="readonly" size="5" value="10%"></p>
   </form>
...
```

Listing 4.11 Datei »form_zahl.htm«, mittlerer Teil

Das Element zur Eingabe eines Zahlenwerts hat den Wert `number` für das Attribut `type`. Setzt man den Cursor in das Eingabefeld, wird eine zusätzliche Up-Down-Schaltfläche zur Änderung der Zahl per Klick angezeigt. Zur Einstellung des Elements können Sie Werte für die Attribute `min`, `max`, `step` und `value` angeben. Diese kennzeichnen die Untergrenze, die Obergrenze, die Schrittweite und den angezeigten Wert. Ein Wert, der nicht innerhalb der Grenzen liegt oder nicht zur Schrittweite passt, wird nicht gesendet. Es erfolgt eine Fehlermeldung, siehe Abbildung 4.24.

Sie können den Wert auch per Code einstellen. Die vier Schaltflächen +1 STEP, –1 STEP, +3 STEP und –3 STEP ändern den Wert um das entsprechende Vielfache der Schrittweite. Die Schaltfläche +3 STEP ändert also z. B. den Wert von 42 in 48. Auch das Attribut `required` kann per Code geändert werden. Markieren Sie die Checkbox, darf das Zahlenfeld nicht mehr leer gelassen werden.

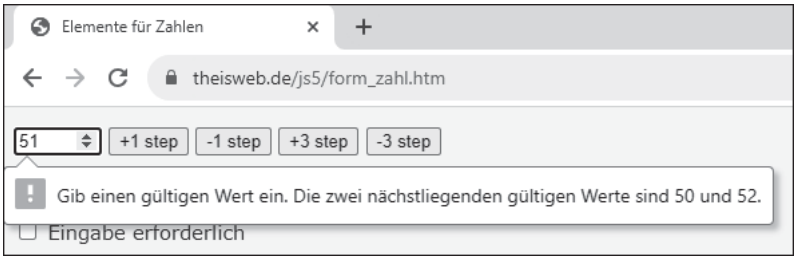

Abbildung 4.24 Nicht zulässige Zahl

Ein Schieberegler (englisch: *slider*) wird zur sicheren Übermittlung eines Zahlenwerts genutzt, der aus einem Bereich stammt. Dieses Element besitzt den Wert `range` für das Attribut `type`. Der Wert selbst wird nicht angezeigt, daher wird hier eine separate Anzeige in einem schreibgeschützten Eingabefeld hinzugefügt. Zur Einstellung gibt es außerdem die Attribute `min`, `max`, `step` und `value`. Es können auch Zahlen mit Nachkommastellen per Schieberegler eingestellt werden.

Der `progress`-Container dient meist zur Anzeige eines prozentualen Fortschritts. Die Attribute `min` und `max` haben die Standardwerte 0 und 1. Hier wird als aktueller Wert 10 und als Wert für das Maximum 100 genommen. Mithilfe der Schaltflächen Start und Stop wird eine JavaScript-Funktion gestartet, die den Fortschrittsbalken animiert. In einem schreibgeschützten Eingabefeld wird der aktuelle Wert angezeigt.

Es folgt der untere Teil des Dokuments:

```
...
   <table>
      <tr><td>Umsatz 2017:</td><td><meter min="0" max="10000"
              value="4500"></meter> 4.500 &euro;</td></tr>
      <tr><td>Umsatz 2018:</td><td><meter min="0" max="10000"
              value="7500"></meter> 7.500 &euro;</td></tr>
      <tr><td>Umsatz 2019:</td><td><meter min="0" max="10000"
              value="6000"></meter> 6.000 &euro;</td></tr>
   </table>

   <script>
      document.getElementById("idForm").addEventListener("submit", senden);

      const zahl = document.getElementById("idZahl");
      document.getElementById("idRequired").addEventListener
         ("click", function() { zahl.required = !zahl.required; } );
```

```
      document.getElementById("idPlus1").addEventListener
         ("click", function() { zahl.stepUp(); } );
      document.getElementById("idMinus1").addEventListener
         ("click", function() { zahl.stepDown(); } );
      document.getElementById("idPlus3").addEventListener
         ("click", function() { zahl.stepUp(3); } );
      document.getElementById("idMinus3").addEventListener
         ("click", function() { zahl.stepDown(3); } );

      document.getElementById("idBereich").addEventListener
         ("change", aendernBereich);
      document.getElementById("idStart").addEventListener
         ("click", startFortschritt);
      document.getElementById("idStop").addEventListener
         ("click", stopFortschritt);
   </script>
</body></html>
```

Listing 4.12 Datei »form_zahl.htm«, unterer Teil

Ein `meter`-Container dient eigentlich zur Anzeige eines Füllungsgrades. Hier werden mithilfe der Attribute `min`, `max` und `value` drei `meter`-Container zur Anzeige von Umsätzen genutzt, ähnlich wie in einem Balkendiagramm.

Mithilfe der Variablen `zahl` wird auf das Zahleneingabefeld verwiesen.

Nach Betätigung der Checkbox wird der Wert der booleschen Eigenschaft `required` mithilfe der logischen Verneinung (Operator `!`) umgedreht. Ist die Checkbox markiert, wird das Zahleneingabefeld zu einem Pflichtfeld und darf nicht leer bleiben.

Die Methoden `stepUp()` und `stepDown()` ändern den Wert des Elements um das entsprechende Vielfache der Schrittweite. Ohne Parameter wird der Wert um genau eine Schrittweite geändert.

Ein Verschieben des Sliders führt zum Ereignis `change` und damit zum Aufruf der Funktion `aendernBereich()`. Es wird dafür gesorgt, dass der aktuelle Wert neben dem Slider steht.

Mit der Schaltfläche START wird die Funktion `startFortschritt()` gestartet, die zu einer Änderung des Fortschrittsbalkens führt. Die Betätigung der Schaltfläche STOP führt zum Aufruf der Funktion `stopFortschritt()`, in der die Änderung wieder beendet wird. Es wird dafür gesorgt, dass der aktuelle Wert neben dem Fortschrittsbalken steht.

Es folgt der JavaScript-Code im ersten Teil des Dokuments:

```
... <head> ...
   <script>
      function senden()
      {
         alert(zahl.value + "\n"
            + document.getElementById("idBereich").value);
      }

      function aendernBereich()
      {
         const bereich = parseFloat(
            document.getElementById("idBereich").value);
         document.getElementById("idBWert").value
            = bereich.toFixed(1);
      }

      let pWert = 10;
      let verweis;

      function startFortschritt()
      {
         pWert++;
         document.getElementById("idFortschritt").value = pWert;
         document.getElementById("idFWert").value = pWert + "%";
         if(pWert<100)
            verweis = setTimeout(startFortschritt, 20);
      }

      function stopFortschritt()
      {
         clearTimeout(verweis);
      }
   </script>
</head>
...
```

Listing 4.13 Datei »form_zahl.htm«, JavaScript-Code

In der Funktion `senden()` wird der Wert des Zahleneingabefelds ausgegeben. Die Funktion `aendernBereich()` dient zur Ausgabe des Zahlenwerts des Schiebereglers mit einer Nachkommastelle.

Die Methode `setTimeout()` des `window`-Objekts führt zu einem zeitlichen Ablauf. Eine Funktion wird zeitverzögert gestartet. Als Parameter dienen ein Verweis auf die betreffende Funktion (hier: `startFortschritt()`) und die gewünschte Zeitverzögerung in Millisekunden. Zurückgeliefert wird ein Verweis. Er wird in der Variablen `verweis` gespeichert, die im gesamten Programm gültig ist. Da sich die Funktion `startFortschritt()` immer wieder selbst aufruft, wird der Fortschrittsbalken vom Startwert 10 in kleinen Schritten bis zum Endwert 100 animiert.

Ein bereits geplanter nächster Aufruf der Funktion `startFortschritt()` kann durch den Aufruf der Funktion `clearTimeout()` verhindert werden. Sie benötigt als Parameter den Verweis, den die Funktion `setTimeout()` zurückgeliefert hat. Damit wird die Animation vorzeitig abgebrochen. Mehr zu zeitlichen Abläufen sehen Sie in Abschnitt 6.5, »Zeitliche Abläufe«.

In der Variablen `pWert`, die im gesamten Programm gültig ist, steht der aktuelle Wert des Fortschrittsbalkens.

4.9.3 Elemente für Zeitangaben

In diesem Abschnitt folgen einige Elemente, die sich zur sicheren Einstellung und Übermittlung von Zeitangaben eignen, siehe Abbildung 4.25. Nach Auswahl eines der Elemente, wie in Abbildung 4.26 zu sehen, erscheinen Bedienelemente zur komfortablen Eingabe oder Auswahl der einzelnen Teile der Zeitangabe innerhalb eines festgelegten Zeitbereichs, wie in Abbildung 4.27.

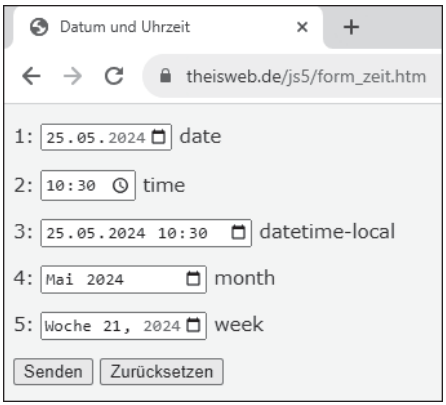

Abbildung 4.25 Elemente für Zeitangaben

Zunächst der Code des Formulars:

```
...
<body>
   <form id="idForm" method="post" action="form_zeit.php">
      <p>1: <input id="idDatum" name="datum" type="date" value="2024-05-25"
         min="2024-04-15" max="2024-06-20"> date</p>
      <p>2: <input id="idUhrzeit" name="uhrzeit" type="time"
         value="10:30" min="08:00" max="13:00"> time</p>
      <p>3: <input id="idLokal" name="lokal" type="datetime-local"
         value="2024-05-25T10:30"> datetime-local</p>
      <p>4: <input id="idMonat" name="monat" type="month" value="2024-05"
         min="2023-10" max="2025-03"> month</p>
      <p>5: <input id="idWoche" name="woche" type="week" value="2024-W21"
         min="2024-W10" max="2024-W30"> week</p>
      <p><input type="submit"> <input type="reset"></p>
   </form>
   <script>
      document.getElementById("idForm").addEventListener("submit", senden);
   </script>
</body></html>
```

Listing 4.14 Datei »form_zeit.htm«, Formular mit Elementen

Alle Elemente haben den Typ `input`, mit unterschiedlichen Werten beim Attribut `type`: `date`, `time`, `datetime-local`, `month` und `week`. Die Attribute `min` und `max` dienen zur Begrenzung des Zeitbereichs. Das Attribut `value` steht für die Voreinstellung der Zeitangabe, jeweils in einem spezifischen Format.

Das erste `input`-Element des Typs `date` dient zur Einstellung eines Datums, mit der Uhrzeit 00:00 Uhr *UTC*. Der Begriff steht für *Universal Time Coordinated*, also für die koordinierte Weltzeit. Sie können einzelne Teile der Zeitangabe zur Eingabe auswählen, siehe Abbildung 4.26.

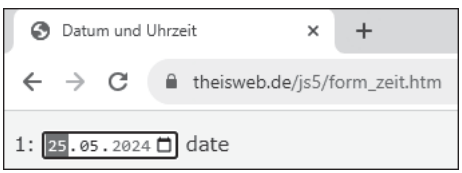

Abbildung 4.26 Auswahl eines Teils der Zeitangabe

Ist kein Datum voreingestellt, erscheint der aktuelle Monat; allerdings wird in diesem Fall beim Senden kein Datum übermittelt. Eine Voreinstellung ist also empfehlenswert. Daten, die außerhalb des erlaubten Bereichs liegen, werden in Hellgrau angezeigt und sind nicht auswählbar, siehe Abbildung 4.27.

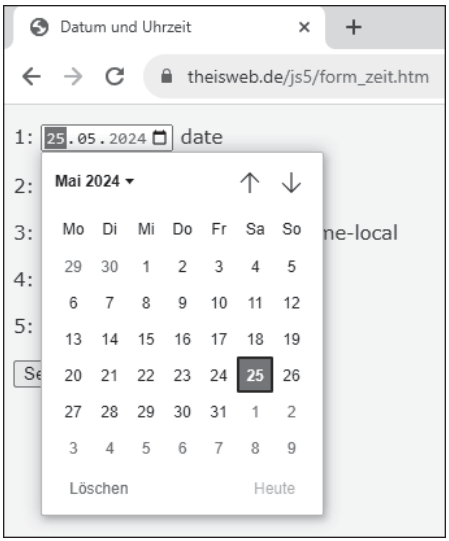

Abbildung 4.27 Begrenzung der Zeitangabe

Das zweite `input`-Element des Typs `time` dient zur Einstellung einer Uhrzeit, mit dem Datum 01.01.1970.

Beim dritten Element des Typs `datetime-local` können sowohl Datum als auch Uhrzeit eingestellt werden. Beachten Sie das gesonderte Format für das Attribut `value`.

Für das vierte Element wird der Typ `month` gewählt. Es können nur ganze Monate eingestellt werden. Das fünfte Element hat den Typ `week`. Es können nur ganze Kalenderwochen eingestellt werden.

Es folgt der Aufbau der Funktion `senden()`:

```
... <head> ...
   <script>
      function senden()
      {
         const d = document.getElementById("idDatum");
         const dWert = d.value;
         const dDatum = d.valueAsDate;
         const dLokal = dDatum.toLocaleDateString();
```

```
        const u = document.getElementById("idUhrzeit");
        const uWert = u.value;
        let uDatum = u.valueAsDate;
        uDatum.setHours(uDatum.getHours() - 1);
        const uLokal = uDatum.toLocaleTimeString();

        const kWert = document.getElementById("idLokal").value;
        const mWert = document.getElementById("idMonat").value;
        const wWert = document.getElementById("idWoche").value;

        alert(dWert + " # " + dLokal + "\n" + uWert + " # " + uLokal
           + "\n" + kWert + "\n" + mWert + "\n" + wWert);
    }
  </script>
</head>
...
```

Listing 4.15 Datei »form_zeit.htm«, JavaScript-Code

Der Wert eines Eingabefelds wird über die Eigenschaft `value` ermittelt.

Bei den ersten beiden Eingabefeldern kann auch ein Wert für die Eigenschaft `valueAsDate` ermittelt werden. Sie enthält den Wert als `Date`-Objekt. Für dieses Objekt können die Funktionen `toLocaleDateString()` und `toLocaleTimeString()` aufgerufen werden, die eine Zeichenkette mit einer lokalen Zeitangabe erzeugen.

Bei der Uhrzeit muss eine Stunde abgezogen werden, damit das angezeigte Ergebnis mit dem eingestellten Wert übereinstimmt. Die Methode `getHours()` liefert den Wert der Stunde eines `Date`-Objekts, die Methode `setHours()` ändert diesen Wert.

Mehr zum `Date`-Objekt finden Sie in Abschnitt 6.4, »Arbeiten mit Zeitangaben«.

In Abbildung 4.28 sehen Sie die Anzeige nach dem Absenden.

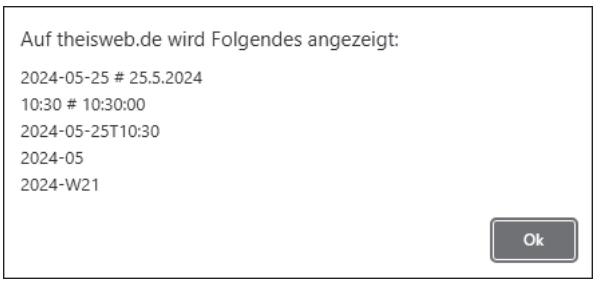

Abbildung 4.28 Ausgabe der Zeitangaben

4.9.4 Validierung von Formularen

JavaScript bietet im Zusammenhang mit der Validierung von Formularinhalten weitere Möglichkeiten. Sie können einstellen, ob ein einzelnes Element bzw. ein ganzes Formular geprüft wird oder nicht. Im Dokument gibt es zwei Formulare jeweils mit einer Schaltfläche SENDEN, siehe Abbildung 4.29.

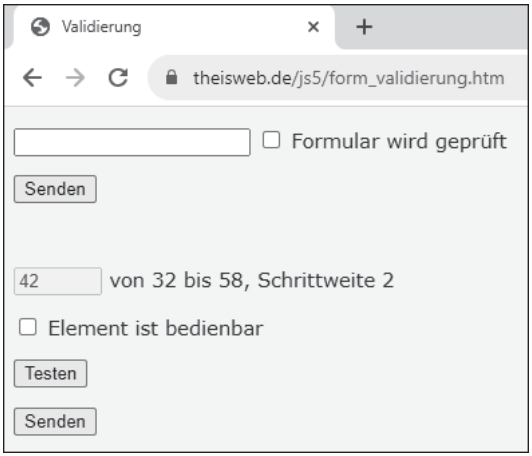

Abbildung 4.29 Zwei Formulare

Das erste Formular enthält ein Eingabefeld für Text. Sie können mithilfe einer Checkbox einstellen, ob das Formular geprüft wird. Das könnte z. B. in Abhängigkeit von eingegebenen Inhalten geschehen.

Beim zweiten Formular können Sie mithilfe einer Checkbox einstellen, ob das darin enthaltene Zahleneingabefeld bedienbar ist oder nicht. Ist es nicht bedienbar, wird die Eingabe nicht geprüft.

Außerdem können Sie durch Betätigung der Schaltfläche TESTEN feststellen, ob die Eingabe geprüft wird und ob sie eine Prüfung bestehen würde. Dies kann sinnvoll sein, falls Inhalte eines Formulars automatisch gesetzt werden und Sie wissen möchten, ob die nunmehr aktuellen Inhalte einer Prüfung standhalten würden.

Es folgt zunächst der untere Teil des Dokuments mit den beiden Formularen und den Eventhandlern:

```
...
<body>
   <form id="idForm1" novalidate method="post" action="form_validierung1.php">
```

```
      <p><input id="idInhalt1" name="inhalt1" required>
      <input id="idCheck1" type="checkbox"> Formular wird geprüft</p>
      <p><input type="submit"></p>
   </form>
   <p> </p>

   <form id="idForm2" method="post" action="form_validierung2.php">
      <p><input id="idInhalt2" name="inhalt2" disabled type="number" min="32"
         max="58" step="2" value="42"> von 32 bis 58, Schrittweite 2</p>
      <p><input id="idCheck2" type="checkbox"> Element ist bedienbar</p>
      <p><input id="idTesten2" type="button" value="Testen"></p>
      <p><input type="submit"></p>
   </form>

   <script>
      document.getElementById("idForm1").addEventListener("submit", senden1);
      document.getElementById("idCheck1").addEventListener("click", check1);

      document.getElementById("idTesten2").addEventListener("click", testen2);
      document.getElementById("idForm2").addEventListener("submit", senden2);
      document.getElementById("idCheck2").addEventListener("click", check2);
   </script>
</body></html>
```

Listing 4.16 Datei »form_validierung.htm«, unterer Teil

Mithilfe des Attributs `novalidate` wird eingestellt, dass das erste Formular nicht geprüft wird. Das Texteingabefeld ist ein Pflichtfeld. Wird es leer gelassen, erscheint dennoch keine Fehlermeldung. Mithilfe der Checkbox kann eingestellt werden, dass das Formular geprüft wird.

Das Zahleneingabefeld des zweiten Formulars ist aufgrund des Attributs `disabled` zunächst nicht bedienbar. Beim Wert 42 würde die Eingabe den Test bestehen. Gültige Werte sind nur die geraden Zahlen von 32 bis 58.

Es folgt der obere Teil des Dokuments:

```
... <head> ...
   <script>
      function senden1()
      {
```

```
      alert(document.getElementById("idInhalt1").value);
   }

   function check1()
   {
      const f = document.getElementById("idForm1");
      f.noValidate = !f.noValidate;
   }

   function testen2()
   {
      const zahl = document.getElementById("idInhalt2");
      alert("Würde geprüft: " + zahl.willValidate + "\n"
         + "Zu groß: " + zahl.validity.rangeOverflow + "\n"
         + "Zu klein: " + zahl.validity.rangeUnderflow + "\n"
         + "Wäre gültig: " + zahl.validity.valid);
   }

   function senden2()
   {
      alert(document.getElementById("idInhalt2").valueAsNumber);
   }

   function check2()
   {
      document.getElementById("idInhalt2").disabled =
         !document.getElementById("idInhalt2").disabled;
   }
</script>
</head>
...
```

Listing 4.17 Datei »form_validierung.htm«, JavaScript-Code

Bei jedem Klick auf die Checkbox des ersten Formulars wird die Funktion `check1()` ausgeführt. Darin wird der Wert der Eigenschaft `noValidate` des Formulars auf `true` bzw. `false` gesetzt. Beim Senden des Formulars erscheint nur dann eine Fehlermeldung wie in Abbildung 4.30, falls `noValidate` den Wert `false` hat und das Eingabefeld leer ist. Sie können also in Ihren Programmen festlegen, ob ein Formular geprüft werden soll oder nicht.

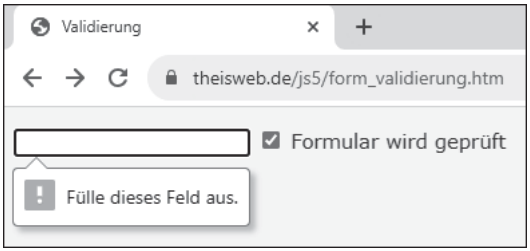

Abbildung 4.30 Nach Wechsel auf »noValidate = false«

In der Funktion `testen2()` werden die Werte einiger boolescher Eigenschaften des Elements bzw. der Eigenschaft `validity` ermittelt:

- `willValidate`: Das Element würde geprüft.
- `validity.rangeOverflow` und `validity.rangeUnderflow`: Der Wert überschreitet bzw. unterschreitet den gültigen Bereich.
- `validity.valid`: Der Wert würde als gültig gesendet.

Der Wert 82 im zweiten Formular führt zu dem Ergebnis in Abbildung 4.31. Die Eigenschaft `rangeOverflow` des `validity`-Objekts liefert in diesem Fall `true`.

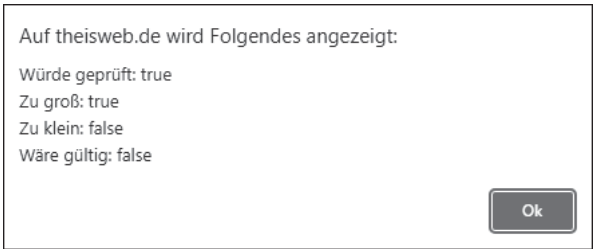

Abbildung 4.31 Testergebnis für den Wert 82

Beim Wert 51 sieht das Ergebnis aus wie in Abbildung 4.32. Der Wert liegt zwar im erlaubten Bereich, ist aber ungerade und daher nicht gültig.

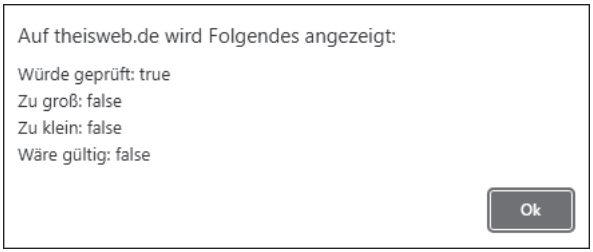

Abbildung 4.32 Testergebnis für den Wert 51

Bei jedem Klick auf die Checkbox des zweiten Formulars wird die Funktion `check2()` ausgeführt. Darin wird der Wert der Eigenschaft `disabled` des Formulars auf `true` bzw. `false` gesetzt. Das Formular kann nur bedient werden, falls `disabled` den Wert `false` hat.

4.10 Dynamisch erstelltes Formular

In diesem Abschnitt wird ein Formular mit JavaScript dynamisch erstellt. Mithilfe einer doppelten `for`-Schleife enthält es anschließend innerhalb einer Tabelle eine größere Anzahl von Eingabefeldern, siehe Abbildung 4.33. Die Werte der Attribute `id` (für JavaScript) und `name` (für PHP) sowie die Eventhandler werden mithilfe von Variablen erstellt.

Die Anzahl der Zeilen und der Spalten kann im Programmcode zu Beginn des Dokuments eingestellt werden. Im vorliegenden Beispiel hat die Tabelle fünf Zeilen und drei Spalten, also 15 Eingabefelder.

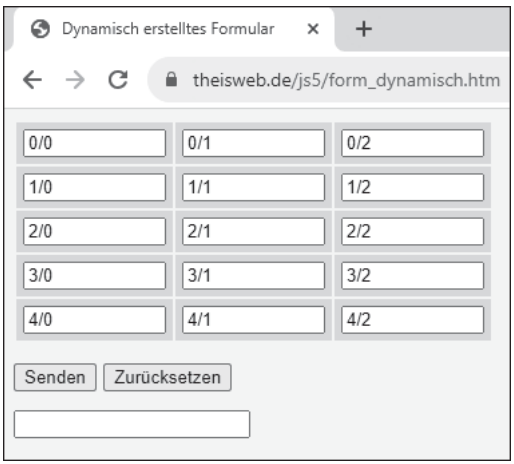

Abbildung 4.33 Tabelle mit Eingabefeldern

Es folgt der Code des Dokuments:

```
... <head> ...
   <script>
      const zeilen = 5, spalten = 3;

      function wechsel(e)
      {
         const id = e.target.id;
         document.getElementById("idReaktion").value
```

```
            = id + ": " +  document.getElementById(id).value;
      }

      function senden()
      {
         let ausgabe = "";
         for(let z=0; z<zeilen; z++)
         {
            for(let s=0; s<spalten; s++)
               ausgabe += document.getElementById("idEin"
                  + z + s).value + " ";
            ausgabe += "\n";
         }
         alert(ausgabe);
      }
   </script>
</head>
<body>
   <form id="idForm" method="post" action="form_dynamisch.php">
   <table>
   <script>
      for(let z=0; z<zeilen; z++)
      {
         document.write("<tr>");
         for(let s=0; s<spalten; s++)
         {
            document.write("<td><input id='idEin" + z + s + "' name='ein"
               + z + s + "' value='" + z + "/" + s + "' size='10'></td>");
            document.getElementById("idEin" + z + s)
               .addEventListener("change", function(e) { wechsel(e); });
         }
         document.write("</tr>");
      }

      document.getElementById("idForm").addEventListener("submit", senden);
      document.write("<input type='hidden' name='zeilen'"
         + " value='" + zeilen + "'>");
      document.write("<input type='hidden' name='spalten'"
         + "value='" + spalten + "'>");
   </script>
```

```
  </table>
  <p><input type="submit"> <input type="reset"></p>
  <p><input id="idReaktion" readonly="readonly"></p>
  </form>
</body></html>
```

Listing 4.18 Datei »form_dynamisch.htm«

Die Anzahl der Zeilen und der Spalten wird zu Beginn des Dokuments in den beiden Variablen `zeilen` und `spalten` festgelegt.

Die beiden Schleifenvariablen `z` und `s` dienen als Bestandteile für die Werte der Attribute `id` und `name`. Beachten Sie die einfachen Hochkommata für die Werte der Attribute innerhalb der Zeichenkette für `document.write()`, die in doppelten Hochkommata steht.

Wird festgestellt, dass sich der Inhalt eines Eingabefelds geändert hat, wird eine anonyme Funktion aufgerufen, die die benannte Funktion `wechsel()` aufruft. Darin wird die `id` des geänderten Eingabefelds ermittelt. Diese wird zusammen mit dem neuen Inhalt des Eingabefelds ausgegeben.

In der Funktion `senden()` werden die Inhalte aller Eingabefelder ermittelt und ausgegeben, wiederum mithilfe einer doppelten `for`-Schleife.

Mithilfe von versteckten Formularelementen wird die Anzahl der Zeilen und der Spalten an die PHP-Datei gesendet. Auf diese Weise können die gesendeten Werte dort ebenfalls dynamisch ermittelt werden. Sie sollten allerdings die Anzahl von 1.000 Eingabefeldern nicht überschreiten, ansonsten meldet PHP in der Standardeinstellung einen Fehler.