

Webcode - Übungsdateien

70F7-5B75-1630

ipso Bildung AG

In Kooperation mit dem HERDT-Verlag stellen wir Ihnen eine PDF inkl. Zusatzmedien für Ihre persönliche Weiterbildung zur Verfügung. In Verbindung mit dem Programm HERDT-Campus ALL YOU CAN READ stehen diese PDFs für Forschung und Lehre nur Mitarbeiterinnen und Mitarbeitern sowie Studierenden der oben genannten Hochschule zur Verfügung. Eine Nutzung oder Weitergabe für andere Zwecke ist ausdrücklich verboten und unterliegt dem Urheberrecht. Jeglicher Verstoß kann zivil- und strafrechtliche Konsequenzen nach sich ziehen.

Isolde Kommer

2. Ausgabe, September 2015

ISBN 978-3-86249-441-5

**HTML5, CSS3 und JavaScript
Webseiten entwickeln**

Fortgeschrittene Anwendungen

HTML5F

Informationen zu diesem Buch	4	5	JavaScript	58
		5.1	Grundlegendes zu JavaScript	58
		5.2	JavaScript in Webseiten einfügen	59
1 Navigation des Webauftritts	6	5.3	Variablen und Datentypen	61
1.1 Die Navigation planen	6	5.4	Operatoren	64
1.2 Navigationsleisten mit HTML und CSS erstellen	8	5.5	Kontrollstrukturen	67
1.3 Navigationsmenüs erstellen	12	5.6	Kommentare	71
1.4 Imagemaps	16	5.7	Übungen	72
1.5 Logische Verlinkung	18			
1.6 Auf eine andere Adresse umleiten	21			
1.7 Übung	22	6	Funktionen und Interaktionen in JavaScript	74
		6.1	Funktionen	74
		6.2	Objektunabhängige Funktionen	80
2 Bereiche definieren und positionieren	24	6.3	Interaktionen	81
2.1 Zeitgemäßes Webseitenlayout	24			
2.2 Container erstellen	25	7	Objektmodell in JavaScript	86
2.3 <code>div</code> -Container und semantische Elemente mit CSS formatieren	27	7.1	Einführung in das Objektmodell	86
2.4 <code>div</code> -Container schweben lassen	28	7.2	Objekt <code>navigator</code>	87
2.5 <code>div</code> -Container positionieren	29	7.3	Objekt <code>window</code>	89
2.6 Das Seitenlayout mit positionierten Elementen aufbauen	31	7.4	Objekt <code>document</code>	90
2.7 Übungen	37	7.5	Objekt <code>event</code>	91
		7.6	Objekt <code>history</code>	95
		7.7	Objekt <code>location</code>	96
		7.8	Vordefinierte Objekte in JavaScript	97
3 Elemente mit CSS anordnen	38	7.9	Übung	106
3.1 Weitergehende Möglichkeiten zur Anordnung von Elementen	38			
3.2 Ebenen anlegen	42	8	Zugriff auf HTML-Elemente	108
3.3 Responsives Webdesign	43	8.1	Grundlagen zum <code>document</code> -Objekt	108
3.4 Übungen	49	8.2	Unterobjekt <code>anchors</code>	109
		8.3	Unterobjekt <code>links</code>	109
		8.4	Unterobjekt <code>applets</code>	110
4 Sound und Videos einbinden	50	8.5	Unterobjekt <code>forms</code>	110
4.1 Multimedia im Web	50	8.6	Eingabefelder	111
4.2 Multimediaformate kennenlernen	51	8.7	Kontroll- und Optionsfelder	112
4.3 Sound in HTML5 einbinden	52	8.8	Auswahllisten	112
4.4 Videos einbinden	54	8.9	Schaltflächen	114
4.5 Übungen	56	8.10	Eingaben überprüfen	114
		8.11	Unterobjekt <code>images</code>	118
		8.12	Übungen	122

9 Dynamic HTML	124
9.1 Bestandteile des dynamischen HTML	124
9.2 DOM-Standard ermitteln	125
9.3 Container erstellen	126
9.4 Ebenen anzeigen und verbergen	126
9.5 Die Position von Ebenen zur Laufzeit ändern	128
9.6 Ebenen bewegen	130
9.7 Umfangreiches Beispiel	132
9.8 Übungen	138
10 JavaScript-APIs der HTML5-Spezifikation	140
10.1 Was ist eine API?	140
10.2 Häufig verwendete JavaScript-APIs	140
10.3 Die Geolocation-API	141
10.4 Der Canvas-2D-Kontext	145
10.5 Text auf den Canvas zeichnen	149
10.6 Übungen	153
11 Kurzeinstieg in AJAX	154
11.1 Was steckt hinter AJAX?	154
11.2 Das <code>XMLHttpRequest</code> -Objekt	155
11.3 Was ist noch möglich?	160
A1 Cascading-Style-Sheet-Referenz	162
A2 JavaScript-Objektreferenz	166
A3 DOM-Referenz	170
Stichwortverzeichnis	171

Informationen zu diesem Buch

Dieses Buch vermittelt Ihnen fundierte Kenntnisse im Umgang mit HTML5, CSS und JavaScript.

Lernziele	Vor jedem Kapitel sehen Sie die jeweiligen Lernziele und Voraussetzungen.
Arbeitsanleitungen	Die effizientesten Wege zum Ziel werden Schritt für Schritt im Kapitel erklärt.
Übungen	Am Ende des Kapitels festigen Sie Ihre erworbenen Kenntnisse.
Referenzen	Im Anhang finden Sie Referenzen zu CSS und JavaScript.

Was bedeuten die Symbole im Buch?



Hilfreiche Zusatzinformation

Praxistipp

Warnhinweis

Nachhaltig lernen mit Beispiel-, Übungs- und Ergebnisdateien

Für die meisten Kapitel stehen Ihnen Beispiel-, Übungs- und Ergebnisdateien zur Verfügung. Alle Dateien sind in nach den Kapiteln benannten Ordnern abgelegt.

- ✓ Mithilfe der **Beispieldateien** können Sie die erläuterten Programmfunctionen direkt nachvollziehen.
- ✓ Die Übungen am Ende der einzelnen Kapitel lassen sich ohne weitere Vorbereitungen mit den jeweiligen **Übungsdateien** durchführen. Hier können Sie auch schnell kontrollieren, ob Sie die Beispiele korrekt ausgeführt haben.
- ✓ Anhand der **Ergebnisdateien** kontrollieren Sie schnell, ob Sie die entsprechenden Übungen korrekt ausgeführt haben.

- Nutzen Sie unsere maßgeschneiderten, im Internet frei verfügbaren Medien. Rufen Sie im Browser die Internetadresse www.herdt.com auf.

1 Wählen Sie Codes.

2 Geben Sie den folgenden Matchcode ein: `HTML5F`.

Was benötigen Sie zum Arbeiten mit diesem Buch?

Um die Arbeitsschritte in diesem Buch durchzuführen, benötigen Sie

- ✓ einen HTML5-fähigen, standardkonformen Webbrowser (z. B. eine aktuelle Version von Chrome, Firefox, Internet Explorer, Safari oder Opera),
- ✓ einen Texteditor, wie zum Beispiel Windows Editor oder die Freeware Notepad++,
- ✓ für einige Beispiele einen aktiven Internetzugang und einen Testserver.

Für die Beispiele im Buch wurden folgende Browser verwendet:

- ✓ Google Chrome 44.0.2403.107
- ✓ Mozilla Firefox 39.0
- ✓ Internet Explorer 10.

Das sollten Sie bereits können

Sie können sich den Kursinhalt am besten aneignen, wenn Sie bereits über folgende Kenntnisse verfügen:

- ✓ Sie besitzen gute Betriebssystem-Kenntnisse (z. B. Windows bzw. OS X).
- ✓ Sie sind sicher im Umgang mit einem Webbrowser.
- ✓ Sie besitzen HTML5- und CSS3-Grundkenntnisse. Wir empfehlen Ihnen dazu das HERDT-Buch *HTML5 - Grundlagen der Erstellung von Webseiten*.

Weitere Medien von HERDT nutzen

Hat Ihnen das vorliegende Buch gefallen, empfehlen wir Ihnen weitere HERDT-Bücher, wie z. B.

- ✓ *CSS - Cascading Style Sheets Level 3 - Grundlagen (CSS3)*
- ✓ *JavaScript 1.8 - Grundlagen (JAVS18)*

Diese und weitere Lehr- und Lernmaterialien können Sie direkt auf unserer Website bestellen: www.herdt.com.

Wir wünschen Ihnen viel Spaß und Erfolg mit diesem Buch.

Ihr Redaktionsteam des HERDT-Verlags

1 Navigation des Webauftritts

In diesem Kapitel erfahren Sie

- ✓ was bei der Planung einer Navigation zu beachten ist
- ✓ wie Sie Text- und grafische Navigationsleisten erstellen
- ✓ wie Sie standardkonforme Navigationsmenüs erstellen
- ✓ wie Sie Imagemaps erstellen
- ✓ wie Sie logische Verlinkungen einsetzen
- ✓ wie Sie auf eine andere Adresse umleiten

1.1 Die Navigation planen

Navigationselemente, Navigationsleisten, Navigationsmenüs

Ein Webauftritt, auch **Website** oder **Site** genannt, besteht aus mehreren miteinander verknüpften HTML-Seiten und eventuell Ressourcen wie PDF-Dokumenten oder downloadbaren Dateien, z.B. ZIP-Archiven.

Diese HTML-Seiten und Ressourcen werden mithilfe von **Navigationselementen**, z. B. Text-Hyperlinks und Buttons, miteinander vernetzt. Sobald der Nutzer auf ein Navigationselement klickt, steuert er die damit verknüpfte Webseite oder andere Ressource an.

Häufig werden die Navigationselemente (Text-Hyperlinks oder Buttons) zu übersichtlichen **Navigationsleisten** oder **Navigationsmenüs** zusammengefasst. Die meisten Webauftritte verfügen über eine oder mehrere solcher Navigationsleisten oder -menüs, mit deren Hilfe sich der Nutzer durch den Webauftritt bewegen kann.



Einstufige horizontale Navigationsleiste mit Text-Hyperlinks



Einstufige horizontale Navigationsleiste mit Buttons



Zweistufiges horizontales Navigationsmenü

Navigationselemente richtig platzieren

Die richtige Platzierung der Gesamtnavigation erleichtert dem Nutzer die Bewegung im Webauftritt. Im Lauf der Zeit haben sich bestimmte Konventionen für die Position der Navigationselemente herausgebildet. Diese Konventionen sollten Sie beachten, damit sich der Nutzer schnell und problemlos in Ihrem Webauftritt zurechtfindet.



Kaum ein Besucher ist bereit, beim Besuch eines Webauftritts zu überlegen, wo er sich gerade befindet und wohin er gelangen kann. Eine solche Website wird er daher bald verlassen.

- ① In der linken oberen Ecke befindet sich ein Link zur Startseite. Meist ist er in Form des Firmen- oder Webauftritts-Logos gestaltet.
- ② Die wichtigsten Sprungverweise werden in Form einer Navigationsleiste oder eines Navigationsmenüs horizontal entlang des oberen Randes erwartet. Möglich ist auch eine vertikale Anordnung entlang des linken Rands.
- ③ Bei stark untergliederten Webseiten wird die horizontale Navigation häufig als Hauptnavigation zu den übergeordneten Themen des Webauftritts verwendet. Die vertikale Navigation dient in diesem Fall als ergänzende Unternavigation, die zu den verschiedenen Seiten des aktuellen Unterthemas führt (siehe nebenstehende Abbildung).
- ④ Mit einer weiteren Navigationsleiste am unteren Seitenrand kann der Nutzer ergänzende Seiten mit FAQ, Kontaktmöglichkeiten, Impressum usw. ansteuern.



Typische Anordnung von Navigationselementen

Der früher häufig genutzte rechte Seitenbereich wird heute nicht mehr für die Hauptnavigation verwendet, weil hier nicht gewährleistet ist, dass die Navigationselemente bei allen Browsetestellungen und Bildschirmgrößen sichtbar sind.



Navigationselemente gestalten



✓ Symbole benennen und verwenden

Navigationselemente sollen möglichst sofort als solche erkennbar und allgemein verständlich sein. Versuchen Sie, eindeutige Hyperlinkbeschriftungen und/oder grafische Symbole zu finden, die möglichst jeder Nutzer versteht.

✓ Ein typisches Beispiel ist das Briefsymbol, das im Web zum Standard für einen E-Mail-Hyperlink geworden ist.



✓ Einheitlichkeit

Damit der Nutzer schon nach kurzer Zeit problemlos durch Ihren Webauftritt navigieren kann, sollten sich die wichtigsten Navigationselemente stets an derselben Stelle der Seiten befinden. Darüber hinaus sollten sie auch auf jeder Seite gleich gestaltet sein, um eine intuitive Orientierung zu ermöglichen. Eine solche konsistente Navigation nennt man persistente oder globale Navigation.

Barrierefreiheit

Bei der Planung der Navigationselemente sollten Sie auch berücksichtigen, auf welche Weise diese von den Suchmaschinen erfasst und von alternativen Ausgabegeräten – wie etwa Screenreadern für sehbehinderte Nutzer – dargestellt werden. Im Idealfall können Ihre Navigationselemente von allen Browsetern, Suchmaschinen und alternativen Ausgabegeräten problemlos erfasst und ausgewertet werden. Man spricht dann von einer vollständigen **Barrierefreiheit**.

Mit Adobe Flash oder JavaScript realisierte Navigationsmenüs sorgen eventuell dafür, dass nur die Startseite Ihres Webauftritts indiziert wird. Manche Ausgabegeräte haben Schwierigkeiten mit der Auswertung solcher Navigationsmenüs. Ähnliches gilt für Imagemaps. Wenn Sie sich für Navigationselemente mit JavaScript, Java oder Flash entscheiden, sollten Sie deshalb für alternative Navigationsmöglichkeiten sorgen, die von allen Ausgabegeräten angezeigt und von den Suchmaschinen erfasst werden können.



Die früher beliebten Flash-Navigationsmenüs haben im modernen Webdesign kaum mehr einen Platz. Die Technologie ist mittlerweile veraltet. Die resultierenden SWF-Dateien werden beispielsweise nicht von iOS unterstützt. Das heißt, dass sie z. B. auf iPhone- und iPad-Geräten nicht nativ angezeigt werden können.

Barrierefreiheit durch Trennung von Struktur und Darstellung

Navigationsleisten und -menüs aus Text- oder grafischen Hyperlinks ohne JavaScript sind barrierefrei. Sie lernen, wie Sie ausschließlich in HTML und CSS erstellte Navigationsleisten und -menüs gestalten. Dabei verwenden Sie für die Strukturierung der Navigation reinen HTML-Code und für die attraktive Darstellung als Navigationsleiste oder -menü CSS-Formatierungen.



Diese strikte Trennung in der Vorgehensweise ist eine Vorbedingung für die problemlose Gestaltung barrierefreier Systeme: Entwickeln Sie zuerst die Inhalte und die Funktionalität bis zum Ende, testen Sie sie und beschäftigen Sie sich danach mit der Gestaltung.

Ein weiterer Vorteil dieser Lösungen ist, dass sie schnell und ohne großen Aufwand anpassbar sind, besonders wenn Sie externe Stylesheets verwenden. Wenn sich das Site-Design ändern soll, müssen Sie nur einige CSS-Definitionen ändern, um für ein neues Erscheinungsbild Ihrer Navigation zu sorgen, sie beispielsweise statt vertikal nun horizontal anzuzeigen.

1.2 Navigationsleisten mit HTML und CSS erstellen

Horizontale Navigationsleiste aus nicht formatierten Text-Hyperlinks erstellen

Die einfachste Möglichkeit, eine Navigationsleiste zu erstellen, ist die Aneinanderreihung von Text- oder grafischen Hyperlinks.

Beispiel: Text-Navigationsleiste ohne Formatierung (*kap01\textnavigation.html*)

Der folgende Code und die zugehörige Abbildung zeigen eine horizontale Navigationsleiste aus Text-Hyperlinks (`<a> ... `):

```
<p><a href="home.html">Home</a> |  
<a href="profil.html">Profil</a> |  
<a href="service.html">Service</a> |  
<a href="jobs.html">Jobs</a> |  
<a href="impress.html">Impressum</a> |  
<a href="hilfe.html">Hilfe</a></p>
```

[Home](#) [Profil](#) [Service](#) [Jobs](#) [Impressum](#) [Hilfe](#)

Horizontale Navigationsleiste mit unformatierten Text-Hyperlinks

Diese einfache Codestruktur genügt als Grundlage für ausgefeilte Darstellungen, die in reinem CSS ohne JavaScript o. Ä. gestaltet werden können:

- ✓ Text-Navigationsleisten, die dynamisch auf die Mausaktionen des Benutzers (Zeigen, Klicken) reagieren
- ✓ unterschiedliche Button-Navigationsleisten, wahlweise auch mit Reaktion auf die Mausaktionen des Benutzers.

Der Vorteil dieser Technik ist, dass solche Navigationselemente schnell geladen werden können und immer funktionieren – selbst in nicht CSS-fähigen oder Nur-Text-Browsern. Dort werden sie als Aneinanderreihung normaler Hyperlinks dargestellt.



In Kapitel 2 lernen Sie eine Möglichkeit kennen, wie Sie Ihren Navigationsbereich mithilfe des `<nav>`-Tags explizit innerhalb Ihrer Seite kennzeichnen.

Text-Navigationsleisten mit CSS gestalten

Um das Aussehen und die Reaktion der oben gezeigten Text-Navigationsleiste *textnavigation.html* zu verbessern, verwenden Sie einfach die **Pseudoklassen** `a:link`, `a:visited`, `a:hover` und `a:active`. Mit diesen können Sie das Aussehen von Text-Hyperlinks während bestimmter und nach bestimmten Mausaktionen des Anwenders ändern, also sogenannte Mouseover-Effekte erzeugen:

Pseudoklassen ermöglichen die CSS-Formatierung von HTML-Elementen, die sich nicht durch ein eindeutiges HTML-Tag ausdrücken lassen. Grundsätzlich geben Sie zuerst das jeweilige Element an, im Beispiel das Tag `<a>` für einen Hyperlink. Anschließend notieren Sie einen Doppelpunkt und eine definierte Angabe wie etwa `link` für einen noch nicht besuchten Hyperlink. Das Aussehen der Pseudoklassen legen Sie wie gewohnt mithilfe von CSS-Eigenschaften fest.



<code>a:link</code>	Der Hyperlink im ursprünglichen Zustand, bevor der Anwender ihn angeklickt oder mit der Maus darauf gezeigt hat
<code>a:visited</code>	Der Hyperlink auf eine bereits besuchte Seite
<code>a:hover</code>	Der Hyperlink beim Hovern (wenn der Benutzer mit der Maus darauf zeigt)
<code>a:active</code>	Der Hyperlink, den der Benutzer gerade anklickt

Achten Sie darauf, bei der Definition der a-Pseudoklassen die in der Tabelle gezeigte Reihenfolge einzuhalten, damit sich die Angaben nicht gegenseitig außer Kraft setzen.



Beispiel: Text-Navigationsleiste mit Mouseover-Effekten (*kap01\mouseover.html*)

Im folgenden Beispiel versehen Sie die zuvor erzeugte einfache Text-Navigationsleiste mit Mouseover-Effekten, sodass sie optisch auf die Aktionen des Benutzers reagiert.

- Öffnen Sie die Beispieldatei *textnavigation.html*.
- Geben Sie direkt vor das schließende `</head>`-Tag den nachfolgend abgedruckten Code ein. Zu Übungszwecken legen Sie die CSS-Stildefinitionen damit in einem eingebetteten Stylesheet fest.

In der Praxis würden Sie statt eines eingebetteten eher ein externes Stylesheet erstellen und im Seitenkopf der einzelnen HTML-Seiten darauf verweisen. Dann könnten Sie bei Bedarf alle Navigationsleisten der Webseiten, in denen auf das externe Stylesheet verwiesen wird, gleichzeitig neu formatieren. Sie müssten dazu nur die Stildefinitionen in der externen Stylesheet-Datei ändern.



```

<style type="text/css">
<!--
① a:link,a:visited,a:hover,a:active {
    font-family:Verdana, Arial, Helvetica, sans-serif;
    font-size:small;
    text-decoration:none;
}
③ a:link {
    color:#339;
}
a:visited {
    color:#909;
}
a:hover {
    color:#C00;
}
a:active {
    color:#999;
}
-->
</style>

```

- ① Zuerst definieren Sie die Eigenschaften, die allen Hyperlink-Zuständen gemeinsam sind: die Schriftfamilie (`font-family`) und die Schriftgröße (`font-size`). Dazu fassen Sie alle vier Pseudoklassen für die einzelnen Zustände des `<a>`-Tags zu einer einzigen Stildefinition zusammen. Die einzelnen Pseudoklassen trennen Sie dabei durch ein Komma.
- ② Mit `text-decoration:none` stellen Sie alle vier Zustände der Hyperlinks im Browser ohne Unterstreichung dar.
- ③ Anschließend notieren Sie die vier Pseudoklassen noch einmal gesondert und geben jeweils die Merkmale an, die sich bei den verschiedenen Hyperlink-Zuständen ändern sollen, im Beispiel die Textfarbe (`color`).



Text-Navigationsleiste mit formatierten Hyperlinks



Die Auswirkungen von `visited` sehen Sie, wenn Sie auf den Link klicken und die Maustaste gedrückt halten.

Button-Navigationsleisten mit CSS gestalten

Die einfache Text-Navigationsleiste im Beispiel `textnavigation.html` kann mit CSS so formatiert werden, dass die einzelnen Hyperlinks das Aussehen von dreidimensionalen Buttons annehmen. Auch dazu verwenden Sie die Pseudoklassen `a:link`, `a:visited`, `a:hover` und `a:active`.

Beispiel: Button-Navigationsleiste (`kap01\3d-navigation.html`)

Im folgenden Beispiel versehen Sie die verschiedenen Hyperlink-Zustände der einfachen Text-Navigationsleiste `textnavigation.html` mit unterschiedlichen Füllungen und Rahmenfarben, sodass sie wie grafische Buttons aussehen, die beim Anklicken eingedrückt erscheinen und ansonsten hervortreten.

- Öffnen Sie die Beispieldatei `textnavigation.html`.
- Löschen Sie die senkrechten Striche (|) und alle Leerzeichen zwischen den einzelnen Hyperlinks.
- Geben Sie innerhalb des Tagpaars `<head> ... </head>` den nachfolgenden Code ein.

```

<style type="text/css">
<!--
① a:link, a:visited, a:hover, a:active {
    border-style:solid;
    border-width:thin;
    font-family:Verdana, Arial, Helvetica, sans-serif;
    font-size:small;
    padding:4px 8px;
    text-align:center;
    text-decoration:none;
    text-transform:uppercase;
}
② a:link {
    background-color:#BDB76B;
    border-color:#F0E68C olive olive #F0E68C;
    color:#000;
}
③ a:visited {
    background-color:#BDB76B;
    border-color:#F0E68C olive olive #F0E68C;
    color:#8B008B;
}
  
```

```

④ a:hover {
    background-color:#F0E68C;
    border-color:olive #F0E68C #F0E68C olive;
    color:#FF8C00;
}
⑤ a:active {
    background-color:#BDB76B;
    border-color:#F0E68C olive olive #F0E68C;
    color:#F0E68C;
}
-->
</style>

```



Button-Navigationsleiste in HTML und CSS

- ① Wie im vorigen Beispiel *mouseover.html* definieren Sie zuerst die Eigenschaften, die sämtlichen Hyperlink-Zuständen gemeinsam sind, beispielsweise die Schriftmerkmale, die Rahmenart (`solid` = durchgezogen) und die Rahmenbreite (`thin`). Für einen entsprechenden Abstand zwischen Text und Buttonkanten sorgt die Eigenschaft `padding`.
- ② Jetzt folgen die Eigenschaften für den Hyperlink im Normalzustand. Mit `background-color` bestimmen Sie die Hintergrundfarbe des Buttons, mit `color` die Textfarbe. `border-color` definiert die Rahmenfarbe: Der linke und der obere Rahmen werden khaki (#F0E68C), der rechte und der untere Rahmen olivfarben. Durch die Helligkeitsunterschiede entsteht der dreidimensionale Eindruck.
- ③ Nach demselben Muster formatieren Sie den besuchten Hyperlink. Ändern Sie nur die Textfarbe (`color`).
- ④ Der Link beim Hovern erhält eine deutlich unterschiedliche Hintergrundfarbe. Ändern Sie auch die Textfarbe. Für die Rahmenfarbe (`border-color`) kehren Sie die Verhältnisse um: Der rechte und der untere Rahmen erhalten das hellere Khaki (#F0E68C), der linke und der obere Rahmen das dunklere Oliv.
- ⑤ Für den aktiven Link wählen Sie eine jeweils andere Hintergrund- und Textfarbe. Die Rahmenfarbe stellen Sie wieder ein wie beim Normalzustand und beim besuchten Hyperlink.

Button-Navigationsleisten mit CSS und Bilddateien gestalten

Bei Bedarf können Sie die zuvor gezeigte Technik zur Gestaltung von Button-Navigationsleisten mit CSS durch Bilder ergänzen. Dadurch ergeben sich weitere Gestaltungsmöglichkeiten, beispielsweise mit Farbverläufen oder Texturen.

Kann das Bild aus irgendeinem Grund nicht geladen werden, schadet dies der Funktionalität der Navigationsleiste nicht. Statt des Bilds wird dann einfach der hellere Hintergrund angezeigt.



Beispiel: Button-Navigationsleiste mit Hintergrundbild (*kap01\3d-navigation-bild.html, burst.jpg*)

Im folgenden Beispiel erhalten die als Buttons dargestellten Hyperlinks beim Hovern ein Hintergrundbild mit einem radialen Farbverlauf.

- ▶ Speichern Sie die Beispieldatei *3d-navigation.html* aus dem vorigen Beispiel und die Beispieldatei *burst.jpg* in demselben Ordner.
- ▶ Öffnen Sie die Datei *3d-navigation.html*.
- ▶ Ändern Sie die Definition der Pseudoklasse `a :hover` folgendermaßen ab:

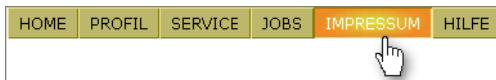
Beispieldatei
"burst.jpg"

```

① a:hover {
    background-color:#F0E68C;
    background-image:url(burst.jpg);
    border-color:olive #F0E68C #F0E68C olive;
    color:#FFFFFF;
}

```

- ① Die Pseudoklasse a :hover wurde durch die Eigenschaft background-image um das Bild burst.jpg ergänzt.



Button-Navigationsleiste mit Hintergrundbild beim Hovern

1.3 Navigationsmenüs erstellen

Nicht nur Navigationsleisten, sondern auch mehrstufige Navigationsmenüs lassen sich in HTML strukturieren und mit CSS gestalten.

Durch diese Möglichkeit können Sie völlig auf JavaScript, Flash und sonstige Technologien verzichten, die häufig zur Erstellung von Navigationsmenüs verwendet werden. Sie stellen damit sicher, dass Ihre Navigationsmenüs sowohl von Suchmaschinen als auch von den alternativen Ausgabegeräten und von Nur-Text-Browsern problemlos ausgewertet werden können.



Horizontales Navigationsmenü

Die HTML-Struktur eines Navigationsmenüs erstellen

Als Grundlage für das Navigationsmenü bietet sich eine unsortierte HTML-Liste an.

Beispiel: HTML-Menüstruktur (*kap01\menustruktur.html*)

```

① <ul>
    <li><a href="#">Über uns</a>
        <ul>
            <li><a href="philos.html">Philosophie</a></li>
            <li><a href="firmeng.html">Firmengeschichte</a></li>
        </ul>
    </li>
    <li><a href="#">Produkte</a>
        <ul>
            <li><a href="buecher.html">Bücher</a></li>
            <li><a href="zeitschr.html">Zeitschriften</a></li>
            <li><a href="software.html">Software</a></li>
        </ul>
    </li>
    <li><a href="#">Kontakt</a>
        <ul>

```

```

<li><a href="beratung.html">Beratung</a></li>
<li><a href="vertrieb.html">Vertrieb</a></li>
</ul>
</li>
</ul>
```

- ① Geben Sie zunächst alle Hauptmenüpunkte als ungeordnete Liste ein.
- ② Diejenigen Punkte, die später im Untermenü erscheinen sollen, geben Sie als untergeordnete Liste ein.
- ③ Um eine vollständige Barrierefreiheit zu erzielen, sollten Sie auch die Hauptpunkte mit Hyperlinks versehen. Denn nur dann lässt sich das Navigationsmenü per -Taste navigieren. Das ist wichtig für eine korrekte Ausgabe auf alternativen Ausgabegeräten. Da die Hauptpunkte keine echten Sprungverweise enthalten, behelfen Sie sich mit Null-Hyperlinks (`href="#"`), Hyperlinks, die zwar als solche erkannt werden, aber nirgendwohin führen.

Von der inhaltlichen Seite her ist das Navigationsmenü fertig. Es funktioniert in allen Browsern und mit sämtlichen alternativen Ausgabegeräten. Sie können sich jetzt mit der Anordnung der Menüpunkte und danach mit der Gestaltung beschäftigen.

- [Über uns](#)
 - [Philosophie](#)
 - [Firmengeschichte](#)
- [Produkte](#)
 - [Bücher](#)
 - [Zeitschriften](#)
 - [Software](#)
- [Kontakt](#)
 - [Beratung](#)
 - [Vertrieb](#)

Fertige Menüstruktur

Das Navigationsmenü mit CSS anordnen

Für die Anordnung der Menüpunkte verwenden Sie CSS.

Beispiel: CSS-Menüanordnung (`kap01\menuanordnung.html`)

Zuerst entfernen Sie die Listenpunkte vor den einzelnen Elementen und definieren Schrift und Positionierung. Mit der Positionierung ordnen Sie die Liste in Menüform an. Erstellen Sie dazu im `head`-Bereich die folgende CSS-Regel:

```

<style type="text/css">
<!--
① ul {
②     font-family: Verdana, Arial, Helvetica, sans-serif;
③     font-size: x-small;
④     list-style: none;
⑤     margin: 0;
⑥     padding: 0;
}
⑦ li {
⑧     float: left;
⑨     position: relative;
⑩    width: 10em;
}
⑪ li ul {
⑫     display: none;
⑬     position: absolute;
⑭     top: 1.2em;
⑮     left: 0;
}
⑯ li:hover ul {
⑰     display: block;
}
-->
</style>
```

- ① Notieren Sie die Definition des Tags `ul` (die unsortierte Liste).
- ② Geben Sie die gewünschten Schriftformatierungen an.
- ③ Als `list-style` wählen Sie `none`, um die Aufzählungspunkte vor den einzelnen Elementen zu entfernen.
- ④ `margin` und `padding` weisen Sie jeweils den Wert `0` zu. Damit bestimmen Sie, dass das Navigationsmenü am linken oberen Rand des Fensters beginnt.
- ⑤ Diese formatierte Liste verwandeln Sie nun in ein horizontales Menü. Dazu definieren Sie das Tag `li` neu, das die einzelnen Listenpunkte erzeugt.
- ⑥ Notieren Sie `float:left` mit einer Breite von `10em` und einer relativen Position. Nur dann werden die verschachtelten Elemente relativ zu den Hauptpunkten angeordnet.



Voll funktionsfähige Navigationsleiste

 Ohne weitere Angaben werden HTML-Elemente im Browser in derselben Reihenfolge wie im Quelltext angezeigt. Mit der CSS-Eigenschaft `float` können Sie Elemente jedoch von Text umfließen lassen. Mit `float:left` steht das Element links und wird rechts umflossen. Detaillierte Informationen über `float` erhalten Sie in Kapitel 2.

 Enthält Ihr Navigationsmenü längere Einträge, erhöhen Sie den Wert `10em` entsprechend.

- ⑦ Als Nächstes definieren Sie die Darstellung der Listenelemente der zweiten Ebene, also der zunächst unsichtbaren Menüelemente. Mit `li ul` erzeugen Sie eine Stildefinition, die nur für innerhalb eines `li`-Tags geschachtelte `ul`-Elemente gilt, also nur für die eingezogenen Unterpunkte.
- ⑧ Setzen Sie `display` auf `none`, damit die eingezogenen Listenelemente zunächst unsichtbar sind.
- ⑨ Positionieren Sie die Elemente anschließend absolut mit der Platzierung `1,2 em` vom oberen und `0` von den übergeordneten Einträgen.
- ⑩ Mit der Pseudoklasse `li:hover` machen Sie das Navigationsmenü schließlich funktionsfähig. Mit `li:hover ul` sprechen Sie nur `ul`-Unterpunkte an, sobald sich die Maus über dem übergeordneten Hauptpunkt befindet. Mit `display:block` bestimmen Sie, dass die jeweiligen Unterpunkte beim Hovern über dem Hauptpunkt angezeigt werden.

Das Navigationsmenü mit CSS gestalten

Das Navigationsmenü ist bisher einfach gehalten. Sie können aber alle von CSS gebotenen Gestaltungsmöglichkeiten einsetzen, um das Navigationsmenü an das Design Ihres Webauftritts anzupassen.

Beispiel: CSS-Menügestaltung (`kap01\menuegestaltung.html`)

```
<style type="text/css">
<!--
ul {
    font-family: Verdana, Arial, Helvetica, sans-serif;
    font-size: small;
    list-style: none;
    margin: 0;
    padding: 0;
}
① li {
    background-color: #999;
    border-bottom-color: #F90;
    border-bottom-style: solid;
    border-bottom-width: .1em;
    border-right-color: #F90;
    border-right-style: solid;
    border-right-width: thick;
}
```

```

        float:left;
        padding:.5em;
        position:relative;
        width:13em;
    }
② li a {
    color:#FFF;
    text-decoration:none;
}
li a[href="#"] {
    font-weight:700;
    text-transform:uppercase;
}
li ul {
    display:none;
    left:0;
    position:absolute;
    top:2.3em;
}

③ li ul a:hover {
    border-bottom-style:solid;
    border-bottom-width:thin;
    border-left-color:#FC0;
    color:#FC6;
}
li:hover ul {
    display:block;
}
-->
</style>
```

- ① Legen Sie im li-Tag die gewünschten Hintergrund- und Rahmenformatierungen fest, sodass die Listenelemente als rechteckige Kästchen dargestellt werden.
- ② Definieren Sie eine Textfarbe (color) für die innerhalb der Listenelemente aufgeführten Hyperlinks und entfernen Sie die standardmäßige Hyperlink-Unterstreichung mittels text-decoration:none.
- ③ Beim Hovern werden die untergeordneten Listenelemente in einer anderen Farbe und mit den entsprechenden Rahmenattributen angezeigt.



Mit CSS formatiertes Navigationsmenü

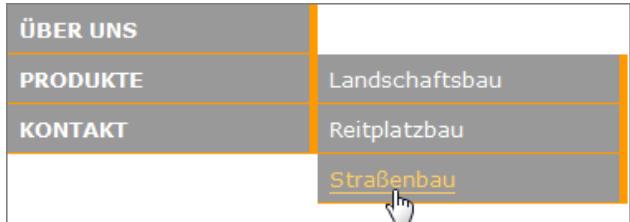
Das horizontale Navigationsmenü mit CSS vertikal ausrichten

Da in der vorliegenden Lösung Inhalt und Gestaltung strikt voneinander getrennt sind, lässt sich das horizontale Navigationsmenü mit wenigen Handgriffen in ein vertikales umwandeln.

Beispiel: Navigationsmenü mit CSS vertikal ausrichten (*kap01\menue-vertikal.html*)

```
<style type="text/css">
<!--
ul {
    /*Definition wie zuvor */
li {
    background-color:#999;
    border-bottom-color:#F90;
    border-bottom-style:solid;
    border-bottom-width:.1em;
    border-right-color:#F90;
    border-right-style:solid;
    border-right-width:thick;
    float:none;
    padding:.5em;
    position:relative;
    width:13em;
}
li a {
    /*Definition wie zuvor */
}
li a[href="#"] {
    /*Definition wie zuvor */
}
li ul {
    display:none;
    left:14.4em;
    position:absolute;
    top:0;
}
li ul a:hover {
    /*Definition wie zuvor */
}
li:hover ul {
    /*Definition wie zuvor */
}
-->
</style>
```

- ① Entfernen Sie `float` aus der `li`-Definition.
- ② In der `li ul`-Definition passen Sie `left` entsprechend der Breite von `li` zuzüglich aller eventuell definierten Rahmenbreiten an.
- ③ Setzen Sie `top` auf 0.



Vertikales Navigationsmenü

1.4 Imagemaps

Was sind Imagemaps?

Als Imagemaps werden Grafiken bezeichnet, in denen bestimmte Bereiche mit einem Hyperlink belegt sind. Diese Bereiche werden **Hotspots** genannt. Hotspots können rechteckig, rund oder polygonförmig (vieleckig) sein. Der Benutzer erkennt den Hyperlink am veränderten Mauszeiger, wenn er diesen über das Bild bewegt.

Imagemaps verwenden

Oft werden Land- bzw. Regionalkarten als Imagemaps angeboten. Der Besucher klickt beispielsweise auf eine Region, und die damit verbundene Seite wird geöffnet.

Früher waren Imagemaps als Navigationselemente äußerst beliebt. Man setzte auf grafisch aufwendig gestaltete Designs und machte sich noch nicht viele Gedanken über die Barrierefreiheit der Seiten. In der heutigen Zeit werden Imagemaps kritisch gesehen, denn sie können nicht von allen alternativen Ausgabegeräten interpretiert werden. Zwar können Sie die Hotspots für eine bessere Zugänglichkeit für die alternativen Ausgabegeräte mit Alternativtexten versehen, doch kommen Textbrowser wie Lynx mit einer herkömmlichen Imagemap nicht klar. Dies ist immer ein deutliches Anzeichen dafür, dass die Seite für ein alternatives Ausgabegerät nicht gut zugänglich ist. Aus diesem Grund sollten Sie Imagemaps möglichst nur als alternative Navigationsmöglichkeit zu einer barrierefreien Hauptnavigation verwenden.



Interaktive Deutschlandkarte

Imagemaps erzeugen

Sie können Imagemaps aus beliebigen Webgrafiken erzeugen. Die als Imagemap dienende Grafik wird wie ein „normales“ Bild mit dem Tag `img` angegeben. Allerdings versehen Sie das Tag `img` mit dem Attribut `usemap`.

Beispiel: Imagemap mit runden Hotspots (`kap01\imagemap.html, schaubild.gif`)

```
<p>
① 
② <map name="schaubild">
③   <area shape="circle" coords="307,123,91" href="tec.html" alt="Technologie">
   <area shape="circle" coords="173,296,97" href="org.html" alt="Organisation">
   <area shape="circle" coords="412,420,148" href="mot.html" alt="Motivation">
</map>
</p>
```

- ① Um eine Imagemap zu erstellen, verwenden Sie das Tag `img` mit dem Attribut `usemap`. Dieses Attribut benötigt als Wert eine URL-Adresse, wozu hier der Ankername `#schaubild` dient.
- ② Mit dem Tag `map` geben Sie das im Ankernamen referenzierte Element an, hier `name="schaubild"`.

Der Name darf keine Leerzeichen enthalten und sollte möglichst nur aus Buchstaben, Ziffern und Unterstrichen (`_`) bestehen.

- ③ Innerhalb des Tagpaars `<map> ... </map>` geben Sie mithilfe von `area`-Tags die Hotspots an. Sie haben hierbei die in der folgenden Tabelle dargestellten Möglichkeiten. Die Koordinaten der Hotspots geben Sie mithilfe des Attributs `coords` in auf die Grafik bezogenen, durch Kommata getrennten Pixelwerten an. Außerdem gehört in das `area`-Tag auch das Verweisziel, das Sie wie bei Hyperlinks üblich mit dem Attribut `href` bestimmen.



Die Imagemap mit dem oben gezeigten Code

<code>area shape="rectangle"</code>	Definiert einen rechteckigen Hotspot. Geben Sie die Koordinaten <code>x1, y1, x2, y2</code> an. <code>x1</code> : linke obere Ecke, von links gemessen <code>y1</code> : linke obere Ecke, von oben gemessen <code>x2</code> : rechte untere Ecke, von links gemessen <code>y2</code> : rechte untere Ecke, von oben gemessen
<code>area shape="circle"</code>	Definiert einen kreisrunden Hotspot. Geben Sie Koordinaten <code>x, y, r</code> an. <code>x</code> : Mittelpunkt, von links gemessen <code>y</code> : Mittelpunkt, von oben gemessen <code>r</code> : Radius in Pixel
<code>area shape="polygon"</code>	Definiert einen vieleckigen (polygonförmigen) Hotspot mit beliebig vielen Ecken. Geben Sie die Koordinaten <code>x1, y1, x2, y2 ... xn, yn</code> an. <code>x</code> : Ecke von links <code>y</code> : Ecke von oben



Die benötigten Koordinaten finden Sie am besten heraus, wenn Sie die Grafik in einem Programm wie etwa Adobe Photoshop öffnen. Wenn Sie mit der Maus über die Grafik fahren, zeigt das Info-Bedienfeld Ihnen die Koordinaten des gerade überfahrenen Bereichs an. Außer Photoshop sind auch Freeware-Programme wie etwa Gimp geeignet. Außerdem gibt es verschiedene Freeware-Programme zum Erstellen von Imagemaps. Beispiele sind etwa Fast Image-Map oder Image Mapper.



Achten Sie darauf, auch bei `<area>`-Tags `<alt>`-Attribute zu verwenden, um die Benutzerfreundlichkeit zu verbessern. Falls die Grafik nicht dargestellt werden kann, werden dann stattdessen diese Texte angezeigt.

1.5 Logische Verlinkung

Hierarchische Beziehungen

Ein Webauftritt besteht nicht einfach aus einer Anhäufung von Webseiten, die nichts miteinander zu tun haben. Die meisten Webseiten haben eine hierarchische Beziehung zueinander oder gehören zu einer Abfolge von Seiten (wie zum Beispiel bei einer Bedienungsanleitung). Zu vielen Websites gehören auch Stylesheet-Dateien, Favicons, Feeds usw.

Ein **Feed** oder **Newsfeed** ist ein über das Web angebotener Nachrichtenstrom, der mit einem Feedreader gelesen werden kann. Der Nutzer kann damit die Inhalte häufig aktualisierter Websites verfolgen, ohne den Webauftritt selbst besuchen zu müssen. Feeds liegen zumeist im **RSS**- oder **Atom-Format** vor.

Ein **Favicon** ist ein Symbol, das im Browser neben dem Dokumenttitel angezeigt wird.

Im head-Bereich Ihrer Webseiten können Sie solche Beziehungen und die zugehörigen Ressourcen, z. B. Stylesheets, notieren. Man spricht dann von einer "logischen Verlinkung". Eines der bekanntesten Einsatzgebiete für die logische Verlinkung ist `<link rel="stylesheet">` zum Einbinden eines externen Stylesheets.



Eines der bekanntesten Einsatzgebiete für die logische Verlinkung ist `<link rel="stylesheet">` zum Einbinden eines externen Stylesheets.

Darstellung logischer Verlinkung

In manchen Browsern – zum Beispiel Opera, iCab, SeaMonkey und Lynx – gibt es Schaltflächen, mit denen bestimmte logische Verlinkungen visualisiert und nutzbar gemacht werden. Geben Sie beispielsweise im Kopf Ihrer Seite eine logische Verlinkung zu einer Impressumsseite an und definieren dabei `copyright` als Beziehung, wird die vordefinierte Copyright-Schaltfläche des Browsers aktiv und bringt den Nutzer nach einem Klick zu dieser Impressumseite.

Im Kopfbereich der Wikipedia-Seite sind beispielsweise die Linktypen `link rel="search"` sowie `link rel="copyright"` definiert und mit den URLs zu den entsprechenden Seiten versehen. Deshalb sind in der Opera-Menüleiste die beiden Schaltflächen *Suchen* und *Copyright* aktiviert.

In Opera zeigen Sie die Menüleiste für die logische Verlinkung mit dem Menübefehl *Ansicht - Symbolleisten - Navigationsleiste* an.



Andere Arten logischer Verlinkung, wie etwa Feeds oder Favicons, werden in der Adresszeile bzw. im Browser-Tab angezeigt.

Auch Suchdienste usw. haben es mit logischen Verlinkungen leichter, die Seitenstruktur Ihres Webauftritts zu erfassen. In Zukunft wird diese Möglichkeit sicherlich an Bedeutung gewinnen.



Syntax des `link`-Elements

<pre><link rel="Linktyp"></pre>	<p><code><link rel</code> definiert im Kopfbereich Ihrer Seite eine logische Beziehung. Danach geben Sie mit <code>rel</code> die Art der Beziehung an. Die in HTML möglichen Beziehungen werden weiter unten in diesem Abschnitt aufgeführt.</p> <pre><link rel="author" href="impressum.htm" title="Impressum"></pre> <p>Mit dem Attribut <code>href</code> geben Sie die URL der zugehörigen Ressource an, mit dem Attribut <code>title</code> optional eine Beschriftung.</p> <p>Weitere Attribute:</p> <ul style="list-style-type: none"> ✓ <code>media</code> bestimmt das Ausgabemedium. ✓ <code>target</code> bestimmt das Zielfenster für das Verknüpfungsziel. ✓ <code>hreflang</code> gibt die Sprache des Verknüpfungsziels an. ✓ <code>charset</code> gibt die Codierung des Verknüpfungsziels an.
---------------------------------------	--

Linktypen für logische Verlinkung

Praxisrelevante Linktypen

Momentan sind nur vier Linktypen für die Praxis von größerer Bedeutung:

<pre><link rel="alternate"></pre>	<p><code>rel="alternate"</code> dient zum Verlinken einer alternativen Darstellungsweise mit demselben Inhalt (beispielsweise ein zugehöriger RSS-Feed innerhalb eines Blogs):</p> <pre><link rel="alternate" type="application/rss+xml" href="rss-feed.xml" title="RSS-Feed"></pre> <p>Weil es sich bei Feeds um XML-Dokumente handelt, wird im obigen Beispiel das <code>type</code>-Attribut mit dem MIME-Typ <code>application/rss+xml</code> verwendet.</p>
---	--

	<p>Die meisten Browser zeigen dann rechts außen in der URL-Adresszeile ein Feedsymbol ① an. Klicken Sie dieses an, können Sie den Feed anzeigen.</p> 
<link rel="icon">	<p>Damit ordnen Sie Ihrer Seite ein Bild (Favicon) zu, das normalerweise im ICO-Format von Microsoft vorliegt:</p> <pre><link rel="icon" type="image/x-icon" href="/favicon.ico" sizes="16x16 32x32"></pre> <p>Geben Sie im Attribut <code>type</code> den MIME-Typ an: <code>image/vnd.microsoft.icon</code>. Sie können aber auch - wie in diesem Beispiel - <code>image/x-icon</code> verwenden.</p> <p>Mit dem Attribut <code>sizes=</code> geben Sie die Größe des Icons an. Das hier verwendete Icon ist in zwei Größen darstellbar: 16 x 16 Pixel und 32 x 32 Pixel. Die beiden Größenangaben werden durch ein Leerzeichen voneinander getrennt.</p>
<link rel="pingback">	<p>Dieses Tag dient beispielsweise im Rahmen einer Blogsoftware der Angabe eines Pingback-Servers:</p> <pre><link rel="pingback" href="http://www.beispiel.de/pingback/xmlrpc"></pre> <p>Ein Pingback erzeugt einen eigenen Artikel zu einem Kommentar eines fremden Artikels (sogar in einem anderen Blog).</p>
<link rel="stylesheet">	<p>Dieses Tag dient zum Verknüpfen einer externen Stylesheet-Datei:</p> <pre><link rel="stylesheet" type="text/css" href="sitetyle.css" title="Mein Sitetyle"></pre> <p>Als MIME-Typ geben Sie <code>type="text/css"</code> an. Denken Sie daran, auch das Attribut <code>title</code> hinzuzufügen, um eine bestmögliche Barrierefreiheit Ihrer Seiten zu erreichen. Dann kann der Nutzer in manchen Browsern zwischen Ihrem und einem eigenen Stylesheet umschalten.</p>



Mit dem **MIME-Typ** (MIME = Multipurpose Internet Mail Extensions) werden die Struktur und der Aufbau von E-Mails festgelegt. Der MIME-Typ wird aber auch zur Deklaration von Inhalten in verschiedenen Internet-protokollen verwendet.

Weitere Linktypen

Außerdem gibt es noch weitere Linktypen für die logische Verlinkung, die in der Praxis aber eher selten verwendet werden. Viele davon aktivieren die entsprechende Schaltfläche im Browser, wenn dieser eine Menüleiste für die logische Verlinkung enthält.

Die folgende Tabelle bietet einen Überblick.

<code><link rel="search"></code>	Verlinkt eine Seite, die eine Suche in der gesamten Website ermöglicht. Nur sinnvoll für Browser, die eine Menüleiste für die logische Verlinkung enthalten.
<code><link rel="author"></code>	Verlinkt eine Seite mit Angaben zum Autor der aktuellen Seite (zum Beispiel dem Impressum). Nur sinnvoll für Browser, die eine Menüleiste für die logische Verlinkung enthalten.
<code><link rel="help"></code>	Verlinkt eine Seite mit Hilfethemen zur aktuellen Seite. Nur sinnvoll für Browser, die eine Menüleiste für die logische Verlinkung enthalten.
<code><link rel="index"></code>	Verlinkt zur Startseite. Nur sinnvoll für Browser, die eine Menüleiste für die logische Verlinkung enthalten.
<code><link rel="first"></code> <code><link rel="last"></code> <code><link rel="next"></code> <code><link rel="prev"></code> <code><link rel="up"></code>	Stellt Verknüpfungen zwischen Seiten einer Serie (<code>first</code> , <code>last</code> , <code>next</code> , <code>prev</code>) oder zu hierarchisch übergeordneten Seiten her (<code>up</code>). Nur sinnvoll für Browser, die eine Menüleiste für die logische Verlinkung enthalten.
<code><link rel="prefetch"></code>	Dieses neue HTML5-Tag verlinkt zur Seite, die der Nutzer mit großer Wahrscheinlichkeit als Nächstes aufrufen wird. Der Browser lädt die Daten dieser Seite, schon während die aktuelle Seite angezeigt wird, sodass die verlinkte Seite im Anschluss deutlich schneller als bisher angezeigt werden kann.
<code><link rel="sidebar"></code>	Verlinkt zu einer Seite mit geeigneten Inhalten für eine Sidebar, zum Beispiel ein Dokument mit einer senkrechten Navigationsleiste. Kann der Browser das Tag interpretieren, kann er das Dokument automatisch in die Sidebar laden.

1.6 Auf eine andere Adresse umleiten

Sie können von der aktuellen Seite automatisch auf eine andere Seite umleiten. Diese Folgeseite wird nach einer festgelegten Zeit geladen und angezeigt. Man spricht auch von **Tunnel-** oder **Doorway-Seiten**.

Eine automatische Weiterleitung sollte nur mit gutem Grund angewendet werden, weil sie die Benutzerfreundlichkeit der Site verringert. Der Nutzer sollte kontrollieren können, was er im Browser darstellen und welche Seite er ansteuern möchte.



Achten Sie darauf, den Nutzer auf der Seite darauf hinzuweisen, in wie vielen Sekunden er weitergeleitet wird. Am besten bieten Sie zusätzlich einen direkten Hyperlink zum Weiterleitungsziel an.

Für die Weiterleitung verwenden Sie das Attribut `refresh` des `meta`-Tags im `head`-Bereich Ihrer Seite:

```
<meta http-equiv="Refresh" content="15;URL=../index-neu.html" />
```

Hier wird die Seite `index-neu.html` nach 15 Sekunden geladen.

Die meisten Suchmaschinen indizieren keine Seiten, deren `refresh`-Wert kleiner als fünf Sekunden ist.



In der Praxis wird diese Technik auch verwendet, um auf die aktuelle Seite weiterzuleiten, diese also zu aktualisieren. So stellen Sie sicher, dass häufig aktualisierte Informationen tatsächlich immer aktuell sind und nicht etwa aus dem Browser-Cache geholt werden.

1.7 Übung

Übung 1: Fragen zur Website-Navigation

Übungsdatei: --

Ergebnisdatei: uebung1.pdf

1. Was versteht man unter Barrierefreiheit?
2. Wo sollte die Hauptnavigation eines Webauftritts angeordnet sein? Warum?
3. Welche Nachteile bieten Imagemaps?

Übung 2: Button erstellen

Übungsdatei: --

Ergebnisdatei: uebung2.html

1. Erstellen Sie den folgenden grauen 3D-Button in reinem HTML und CSS. Beim Hovern soll sich die Farbe der Buttonfläche in ein helles Gelb ändern. Verwenden Sie die folgenden CSS-Eigenschaften: text-decoration, padding, font-weight, border-width, border-style, border-color, color, background-color.

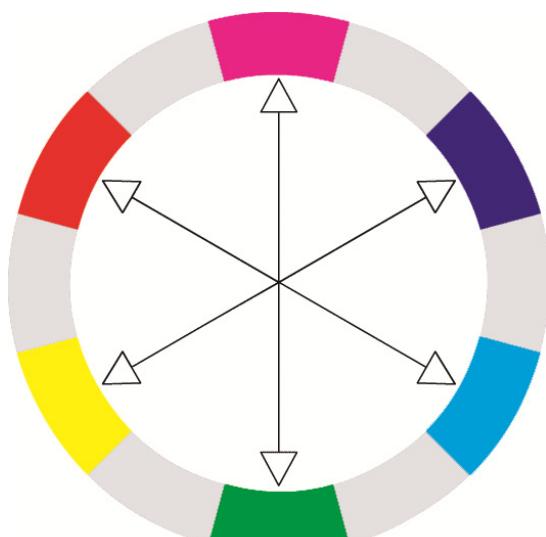


Übung 3: Imagemap erstellen

Übungsdatei: farbkreis.gif

Ergebnisdatei: uebung3.html

1. Fügen Sie das Bild *farbkreis.gif* in eine HTML-Seite ein.
2. Versehen Sie jede der sechs Farben mit einem polygonförmigen Hotspot.
3. Weisen Sie den Hotspots die Verweisziele *rot.html*, *magenta.html*, *blau.html*, *cyan.html*, *gruen.html* und *gelb.html* zu.
4. Statten Sie sie mit den entsprechenden Alternativtexten aus.



Grundlage für die zu erstellende Imagemap

Übung 4: Fragen zur logischen Verlinkung

Übungsdatei: --

Ergebnisdatei: *uebung4.pdf*

1. Wie ordnen Sie Ihrer Seite ein Favicon zu?
2. Woran erkennt der Anwender, dass einer Seite mit `<link rel="alternate">` ein Feed zugeordnet wurde?
3. Welches HTML-Element nutzen Sie, um eine bestimmte Seite im Voraus zu laden, während die aktuelle Seite angezeigt wird?

2 Bereiche definieren und positionieren

In diesem Kapitel erfahren Sie

- ✓ wie Sie div-Container einsetzen
- ✓ wie Sie Bereiche mit semantischen Tags definieren
- ✓ wie Sie Container positionieren

2.1 Zeitgemäßes Webseitenlayout

Nachteile von Tabellen als Layoutwerkzeug

Bis vor wenigen Jahren war es in der Welt des Web-Designs allgemein üblich, Seitenelemente mithilfe von unsichtbaren Tabellen an bestimmten Stellen der Seite zu fixieren. In rein visueller Hinsicht waren Tabellen für das Seitenlayout auch recht gut geeignet, hatten aber schwerwiegende Nachteile:

- ✓ Für ein stabiles Seitenlayout mussten transparente GIF-Bilder verwendet werden, die die Breiten und Höhen der unsichtbaren Tabellen und damit die Abstände der Seitenlayoutelemente fixierten.
- ✓ Bilder mussten häufig im Bildbearbeitungsprogramm zerstückelt (in „Slices“ unterteilt) und in HTML wieder wie ein Puzzle zusammengesetzt werden.
- ✓ Für verschiedene Browser waren teilweise einander widersprechende Tabellenformatierungs-Tags notwendig.
- ✓ Durch ständig wiederholte `table` und `td`-Tags wurde der Code unnötig aufgebläht.
- ✓ Tiefgreifende Umgestaltungen des Seitenlayouts waren schwierig und aufwändig.
- ✓ Struktur und Darstellung der Seite befanden sich komplett im HTML-Code.

Vorteile von Containern als Layoutwerkzeug

Aus diesem Grund hat das W3C die Empfehlung ausgesprochen, HTML-Tabellen nicht für Layoutzwecke zu verwenden, sondern ausschließlich für die Strukturierung von Tabellendaten.

Für das Seitenlayout hingegen verwenden Sie am besten mit CSS formatierte **div-Container** und ggf. die neuen HTML5-Tags für Kopf- und Fußzeilen, Artikel usw. Diese Herangehensweise bietet folgende Vorteile:

- ✓ vollständige Trennung von Struktur und Darstellung,
- ✓ bessere Kontrolle über das Seitenlayout,
- ✓ schnelle und einfache Aktualisierung oder Umgestaltung.



Diese Herangehensweise bedeutet auch, dass Sie verschiedene Stylesheets für verschiedene Zwecke spezifizieren können, ohne mehrere Versionen Ihrer Website verwalten zu müssen. Damit können Sie verschiedene Versionen des Webauftritts für unterschiedliche Ausgabegeräte wie etwa Smartphones, Tablets und sogar Screenreader für sehbehinderte Nutzer erzeugen. Auch eine spezielle Druckversion lässt sich ohne allzu großen Aufwand realisieren: Sie benötigen dazu nur ein Stylesheet, z. B. mit geeigneter Farbe und Seitenrändern.

2.2 Container erstellen

Bereiche mit `div`-Containern definieren

Einzelne Gestaltungseinheiten bzw. Layout-Bereiche, die Sie an verschiedenen Stellen der Seite verteilen möchten, können Sie durch `div`-Elemente voneinander trennen.

<code>div</code>	<p><code>div</code> leitet sich vom englischen "division" ("Bereich") ab und bezeichnet einen gemeinsamen Bereich, in den Sie mehrere Inhalte aller Art einschließen können, beispielsweise Texte, Grafiken, Datentabellen usw.</p> <p>Wenn Sie einen <code>div</code>-Container nicht mit Cascading Stylesheets formatieren, beginnen alle in den Container eingeschlossenen Elemente in einer neuen Zeile.</p>
------------------	--

Semantische HTML5-Elemente anstelle von `div`-Containern verwenden

HTML5 sieht die Verwendung des `div`-Elements nur dann vor, wenn es kein anderes Element gibt, mit dem Sie den gewünschten Bereich exakter auszeichnen können. Es gibt **semantische HTML5-Elemente** für Elemente wie Kopf- und Fußzeilen, Artikel, Seitenleisten und Abschnitte. Diese Tags funktionieren grundsätzlich wie `div`-Container, sorgen jedoch – richtig eingesetzt – für ein semantisch korrektes Dokument.



<code>aside</code>	Dient zur Definition einer Seitenleiste. Innerhalb des Tagpaars <code><aside> ... </aside></code> können sämtliche verfügbaren Block- und Inline-Elemente verschachtelt werden.
<code>article</code>	Dient zur Definition eines Artikels, beispielsweise in einem Magazin oder Blog. Innerhalb des Tagpaars <code><article> ... </article></code> können Sie sämtliche verfügbaren Block- und Inline-Elemente verschachteln.
<code>figure</code>	Dient zur Definition eines Abbildungsbereichs: Sie können damit Bilder und Bildunterschriften miteinander gruppieren
<code>figCaption</code>	Dient zur Definition einer Abbildungsunterschrift. In einem <code>figure</code> -Element (siehe oben) werden die Bilder über das <code>img</code> -Element definiert, die Bildbeschriftungen über das <code>figCaption</code> -Element.
<code>footer</code>	Dient zur Definition eines Fußbereichs, beispielsweise für die Zusatznavigation usw. Setzen Sie den Fußbereich zwischen das Tagpaar <code><footer> ... </footer></code> . Anders als bei <code>div</code> -Tags können Sie in das <code>footer</code> -Tag keine weiteren <code>footer</code> verschachteln.
<code>header</code>	Dient zur Definition eines Kopfbereichs, wie etwa das Firmenlogo, die waagerechte Hauptnavigation, den Seitentitel etc. Setzen Sie den Kopfbereich zwischen das Tagpaar <code><header> ... </header></code> . Anders als bei <code>div</code> -Tags können Sie in das <code>header</code> -Tag keine weiteren <code>header</code> verschachteln.
<code>main</code>	Dient zur Definition des Hauptinhalts eines Dokuments. Der Inhalt innerhalb des <code><main></code> -Elements sollte im Dokument nur ein einziges Mal vorkommen, also keine mehrfach verwendeten Navigationslinks, Copyright-Informationen, Logos usw. enthalten.
<code>nav</code>	Dient zur Definition eines Navigationsbereichs. Innerhalb des Tagpaars <code><nav> ... </nav></code> können sämtliche verfügbaren Block- und Inline-Elemente verschachtelt werden.

section	Dient zur Definition eines Abschnitts, besonders in umfangreichen und stark strukturierten HTML-Elementen. Innerhalb des Tagpaars <section> ... </section> können sämtliche verfügbaren Block- und Inline-Elemente verschachtelt werden, beispielsweise auch article-Elemente.
template	Dient zur Gruppierung von Inhalten, die nicht beim Laden der Seite, sondern anschließend zur Laufzeit mittels JavaScript dargestellt werden.

Vorgehensweise beim Gestalten des Seitenlayouts mit div-Containern oder semantischen HTML-Elementen

Um ein Seitenlayout mit div-Containern oder semantischen HTML-Elementen aufzubauen, können Sie beispielsweise so vorgehen wie in der folgenden Tabelle aufgeführt. Nach dieser Reihenfolge richtet sich auch der Aufbau dieses Kapitels.

Container erstellen	Zunächst fertigen Sie eine Skizze des gewünschten Seitenlayouts an, z. B. im Bildbearbeitungsprogramm oder auf einem Blatt Papier. Erstellen Sie anschließend in HTML die benötigten div-Container oder semantischen HTML-Elemente. Für jeden Seitenbereich – etwa Hauptnavigation, Inhaltsbereich – erstellen Sie ein eigenes Element.
Container formatieren	Weisen Sie den Elementen per Cascading Style Sheets (CSS) die gewünschten Formatierungsmerkmale wie Hintergrundfarbe, Rahmen, Schriftmerkmale zu. Dabei definieren Sie für jedes Element die benötigte Breite und eventuell eine Höhe.
Container positionieren	Positionieren Sie die einzelnen Elemente an der gewünschten Stelle der Seite.

Beispiel: Zwei Container ohne CSS-Formatierung (*kap02\div-container.html*)

<pre> ① <nav> Home Aktuelles Unser Angebot Kontakt Impressum </nav> ② <div> <h1>Aktuelles: Ein interessanter Artikel</h1> Interessanter Artikeltext </div> </pre>

Hier wurde ein nav-Element für die Navigation ① und ein div-Container für den Seiteninhalt ② erstellt.

Da es keine CSS-Formatierungen gibt, werden die beiden Elemente untereinander dargestellt, wie die nebenstehende Abbildung zeigt.



Die Elemente können auch ineinander verschachtelt werden. Ein Beispiel erhalten Sie in Abschnitt 2.6.

The screenshot shows a browser window with the following content:

- Aktuelles** (highlighted in blue)
- Home
- Aktuelles
- Unser Angebot
- Kontakt
- Impressum

Aktuelles: Ein interessanter Artikel

Interessanter Artikeltext

Unformatierte Container

2.3 div-Container und semantische Elemente mit CSS formatieren

Die in Ihrem Dokument definierten div-Bereiche können Sie mit CSS gestalten.

Beispiel: Zwei Container mit CSS-Formatierung (*kap02\div-container-formatiert.html*)

- Öffnen Sie die Beispieldatei *div-container.html*.
- Definieren Sie innerhalb des Tagpaars `<head> ... </head>` die folgenden CSS-Stildefinitionen. Sie versehen die Seite dadurch mit einem eingebetteten Stylesheet:

```
① <style type="text/css">
#Nav {
    font-family: Arial, Helvetica, sans-serif;
    font-size: small;
    background-color: #FFC;
    border: 2px dotted #600;
}
② #Artikel {
    border: 3px double #600;
    padding: 3px;
}
</style>
```

- ① Erstellen Sie eine ID für die Navigation. Weisen Sie der ID die gewünschten CSS-Formatierungen zu.
- ② Erstellen Sie eine ID für den Seiteninhalt, und weisen Sie der ID die gewünschten CSS-Formatierungen zu.

```
③ <nav id="Nav">
    <ul>
        <li>Home</li>
        <li>Aktuelles</li>
        <li>Unser Angebot</li>
        <li>Kontakt</li>
        <li>Impressum</li>
    </ul>
</nav>
④ <div id="Artikel">
    <h1>Aktuelles: Ein interessanter Artikel</h1>
    <p>Interessanter Artikeltext</p>
</div>
```

- ③ Im body-Bereich des Dokuments weisen Sie dem nav-Element die ID Nav zu.
- ④ Weisen Sie dem div-Container die ID Artikel zu.

Da die jeweiligen Elemente nur ein einziges Mal auf der Seite vorkommen, eignen sich IDs für die Formatierung. Wenn die Elemente auf der Seite mehrfach vorkommen, verwenden Sie stattdessen Klassen.

The screenshot shows a web page with a yellow header containing a navigation menu with links to Home, Aktuelles, Unser Angebot, Kontakt, and Impressum. Below the header is a red-bordered box containing the heading "Aktuelles: Ein interessanter Artikel" and the text "Interessanter Artikeltext". A small orange info icon is in the bottom right corner of the page.

Containern eine Breite und eine Höhe zuweisen

Wenn Sie nichts anderes angeben, nehmen div-Container stets die gesamte Breite der Webseite oder des übergeordneten Containers ein. Dieses übergeordnete Element kann auch als **Eltern-Element** bezeichnet werden.

Damit Sie einen Container frei auf der Seite positionieren oder schweben lassen können (vgl. Abschnitt 2.4 und 2.5), müssen Sie per CSS auch seine Breite definieren.

width: Wert	Mit der CSS-Eigenschaft <code>width</code> können Sie die Breite des Elements festlegen. Für die Angabe des Werts haben Sie folgende Möglichkeiten: <ul style="list-style-type: none"> ✓ Zahlenangaben mit entsprechender Maßeinheit, ✓ prozentuale Angaben relativ zur Breite des Eltern-Elements, ✓ <code>auto</code> = automatische Festlegung.
height: Wert	Neben der Breite eines Elements können Sie auch dessen Höhe bestimmen. Die Wertangabe entspricht der für die Eigenschaft <code>width</code> .

2.4 div-Container schweben lassen

Der normale Elementfluss sieht vor, dass die Container in der Reihenfolge, wie sie im Quelltext auftauchen, aufeinander folgen. Das heißt, die Absätze werden untereinander dargestellt. Mit der CSS-Eigenschaft `float` können Sie div-Container und semantische HTML-Elemente jedoch von Text umfließen lassen.

float: Wert	Die folgenden Werte können Sie für <code>float</code> angeben: <ul style="list-style-type: none"> ✓ <code>left</code>: Element steht links und wird rechts umflossen. ✓ <code>right</code>: Element steht rechts und wird links umflossen. ✓ <code>none</code>: Element wird nicht umflossen. Die Eigenschaft <code>float</code> lässt sich nicht auf absolut positionierte Elemente (vgl. Abschnitt 2.6) anwenden.
--------------------	--

Damit die Eigenschaft `float` eine sichtbare Auswirkung hat, müssen Sie für den div-Container eine Breite festlegen (z. B. 50 %). Nur wenn neben dem Element Platz ist, können daneben weitere Inhalte stehen.

Das umfließende Element findet sich nicht neben dem Element mit der Eigenschaft `float`, sondern dahinter. Nur die Inhalte des Absatzes werden verschoben: Durch `float` wird der entsprechende Platz komplett frei, die nachfolgenden Elemente rutschen nach oben, die darin enthaltenen Texte und Bilder werden in den sichtbaren Bereich gerückt.

Beispiel: Schwebender div-Container ([kap02\div-container-schwebend.html](#))

In diesem Beispiel definieren Sie einen div-Container mit Textinhalt. Sie lassen ihn dann über dem Rest des Dokumentinhalts schweben.

<pre><!doctype html> <html> <head> <meta charset="utf-8"> <style type="text/css"> <!-- ① #rahmen { width:15em; float:left; font-family:Verdana, Arial, Helvetica, sans-serif; font-size:x-small; background-color:#FFC; border:1px solid #036;</pre>	
---	--

```

        margin-right:2em;
        padding-left:1em;
    }
    #absatz {
        background-color:#0FF;
    }
    -->
</style>
<title>Float links</title>
</head>
<body>
② <div id="rahmen">
    <p>Dieser Container besitzt die Eigenschaft "float:left;"</p>
</div>
③ <p id="absatz">Dieser Text wird durch das links schwebende
("float:left;") Element nach rechts verschoben.</p>
</body>
</html>

```

- ① Erstellen Sie im head des Dokuments eine ID für Ihren div-Container. Weisen Sie der ID die gewünschten CSS-Formatierungen inklusive der CSS-Eigenschaft float:left zu.
- ② Erstellen Sie im body des Dokuments einen div-Container, dem Sie die zuvor definierte ID zuweisen.
- ③ Geben Sie nach dem div-Container weitere Inhalte ein.

Dieser Container besitzt
die Eigenschaft
"float:left;"

Dieser Text wird
durch das links
schwebende
("float:left;") Element nach rechts gedrückt.

Über dem Seiteninhalt schwebender div-Container

2.5 div-Container positionieren

Für die Positionierung gibt es seit der Version CSS2 (Cascading Style Sheets Level 2) über die CSS-Eigenschaft position vier verschiedene Möglichkeiten, ein Element zu positionieren. Sie können damit Elemente unabhängig von ihrer Position im Quelltext an einer beliebigen Stelle des Browserfensters anzeigen.

position: Wert	Folgende Werte stehen zur Verfügung: <ul style="list-style-type: none"> ✓ static: keine spezielle Formatierung; Element positioniert sich im Textfluss ✓ relative: Abweichung von der Position, die das Element im Textfluss eingenommen hätte ✓ absolute: Positionierung des Elements mit Bezug auf das Element body oder den letzten Vorfahren mit einer relativen Positionierung ✓ fixed: wie absolute, aber beim Scrollen der Webseite bleibt das Element fixiert
-----------------------	---

Wo der div-Container genau auf der Seite platziert werden soll, geben Sie über die vier Startpositionen an. Sie können hier Zahlenwerte mit der entsprechenden Maßeinheit oder prozentuale Angaben zur Höhe und Breite des Eltern-Elements verwenden. Mit auto stellen Sie eine automatische Positionierung ein.

left: Wert	Abstand vom linken Rand des Containers
top: Wert	Abstand vom oberen Rand des Containers
right: Wert	Abstand vom rechten Rand des Containers
bottom: Wert	Abstand vom unteren Rand des Containers



- ✓ Alle Dezimalstellen der Positionsangaben müssen mit einem Dezimalpunkt abgetrennt werden, beispielsweise `top: 3.2cm` und `left: 6.2px`.
- ✓ Haben Sie ein Element mit `position: static` oder keiner Positionsangabe zwischen die anderen Elemente fließen lassen, sind die Angaben der Eigenschaften `left`, `top`, `right` und `bottom` unwirksam.

Absolute Positionierung

Die absolute Positionierung bezieht sich auf ein Koordinatensystem, das über das HTML-Dokument gelegt wird. Der Nullpunkt liegt in der linken oberen Ecke. Von hier ab wird nach links (x-Achse) und nach unten (y-Achse) gemessen.

Beispiel: Absolut positionierter div-Container ([kap02\div-container-absolut.html](#))

Der folgende Code zeigt ein absolut positioniertes Element. Dieses stellt innerhalb der Seite eine absolut unabhängige Einheit dar, die in der Grundeinstellung einfach über allen anderen Seitelementen platziert wird. Das Element wird also vollständig aus dem Fluss des Dokuments herausgenommen. Die übrigen Seitelemente verhalten sich, als sei das Element nicht vorhanden.

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
① <style type="text/css">
<!--
#rahmen {
    position: absolute;
    width: 250px;
    background-color: #FFC;
    border: 1px solid #036;
    height: auto;
    left: 10px;
    padding: 5px;
    top: 10px;
}
-->
</style>
<title>Absolute Positionierung</title>
</head>
<body>
② <p>Dieser Text hat mit dem positionierten Rahmen nichts zu tun. Er wird einfach hinter diesem dargestellt. </p>
③ <div id="rahmen">
    <p>Dieser Rahmen ...</p>
</div>
</body>
```

- ① Erzeugen Sie eine CSS-Stildefinition für den zu positionierenden div-Container. Nehmen Sie in diese die absolute Positionierung mit auf (`position: absolute`).
- ② Geben Sie zur Verdeutlichung einen beliebigen `<p>`-Absatz ein.
- ③ Erzeugen Sie einen div-Container, den Sie mit der zuvor erstellten ID formatieren.

Dieser Rahmen ist absolut positioniert. Er liegt über allen anderen Seitelementen.

Absolut positionierter div-Container

Relative Positionierung

Die absolute Positionierung ist nur dann sinnvoll, wenn Sie akzeptieren, dass der Bezugspunkt des Elements der Seitenrand bzw. der letzten Vorfahre mit einer relativen Positionierung ist. Für den Aufbau eines differenzierten Layouts ist die absolute Positionierung deshalb nur bedingt geeignet, weil dabei Elemente nicht nur an der Seite, sondern auch aneinander ausgerichtet werden.

Hier bietet sich eine relative Positionierung an. Das relativ positionierte Element wird nicht am HTML-Dokument selbst, sondern an seiner Position im HTML-Dokument ausgerichtet. Die Position wird ab der tatsächlichen Platzierung des `div`-Containers gemessen.

Beispiel: Relativ positionierter `div`-Container (*kap02\div-container-relativ.html*)

Anhand eines praktischen Beispiels lässt sich dies schnell erfassen.

- ▶ Öffnen Sie die Beispieldatei *div-container-absolut.html*.
- ▶ Ändern Sie die Position in der Stildefinition für den Rahmen im obenstehenden Beispiel ab.

```
#rahmen {
    position: relative;
    /* ... */
}
```

Nun befindet sich der Ausgangspunkt für die Berechnung der Position des Rahmens nicht mehr in der linken oberen Dokumentecke, sondern unter dem vorhergehenden `p`-Absatz, weil der `div`-Container unter diesem Absatz eingefügt wurde.

Dieser Text hat mit dem positionierten Rahmen nichts zu tun. Er wird einfach hinter diesem dargestellt.

Dieser Rahmen ist absolut positioniert. Er liegt über allen anderen Seitenlementen.

Relative Positionierung

2.6 Das Seitenlayout mit positionierten Elementen aufbauen

Mit den vorgestellten Methoden zur Positionierung können Sie nun Ihre Seitenlayouts aufbauen. Nachfolgend sehen Sie drei Beispiele.

Beispiel: Seitenlayout mit absoluter und relativer Positionierung (*kap02\div-layout.html*)

Mithilfe eines `div`-Containers und semantischen Elementen soll das nebenstehend abgebildete Seitenlayout entstehen.

- ① Links soll eine senkrecht angeordnete Navigationsleiste zu sehen sein.
- ② Rechts daneben soll sich der Inhaltsteil der Seite befinden,
- ③ darunter eine Fußzeile.

①	<ul style="list-style-type: none"> • Link 1 • Link 2 • Link 3 	<p>Der Inhalt kann beliebig lang werden; die Position der Fußzeile passt sich wegen ihrer relativen Positionierung stets an die Länge des Inhaltskastens an.</p>
②	<p>Der Inhalt kann beliebig lang werden; die Position der Fußzeile passt sich wegen ihrer relativen Positionierung stets an die Länge des Inhaltskastens an.</p>	<p>Fußzeile ③</p>

Geplantes Seitenlayout

- Die Navigationsleiste kann absolut positioniert werden, weil sie sich immer an derselben Stelle des Dokuments befindet.
- Der Inhaltsteil hingegen muss eine relative Positionierung erhalten. Seine Länge kann unterschiedlich sein, und die Position der Fußzeile hängt von dieser Länge ab.
- Die Fußzeile muss ebenfalls relativ positioniert sein, weil sie unterhalb des Inhaltsteils mit seiner variablen Länge liegen soll.

```

<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>DIV-Layout</title></head>
<style type="text/css">
<!--
① body { margin: 0 }
#navigation {
②     position: absolute;
     padding: 5px;
     width: 20%;
}
③ #inhalt {
     position: relative;
     padding: 5px;
     width: 80%;
     left: 20%;
}
#fusszeile {
     position: relative;
     padding: 5px;
     width: 80%;
     left: 20%;
}
-->
</style>
</head>

<body>
<nav id="navigation">
    <ul>
        <li>Link 1 </li>
        <li>Link 2</li>
        <li>Link 3</li>
    </ul>
</nav>
<div id="inhalt">
    <p>...</p>
④ </div>
<footer id="fusszeile"> Fußzeile </footer>
</body>
</html>

```

Im abgedruckten Code wurden einige Formatierungseinstellungen weggelassen, damit der Code nicht zu lang wird. In der Beispieldatei *div-layout.html* ist der gesamte Code vorhanden.

- ① Damit die Container für Inhalt und Navigation auf der Y-Achse an derselben Stelle beginnen, versehen Sie das body-Tag mit der Eigenschaft `margin:0`.

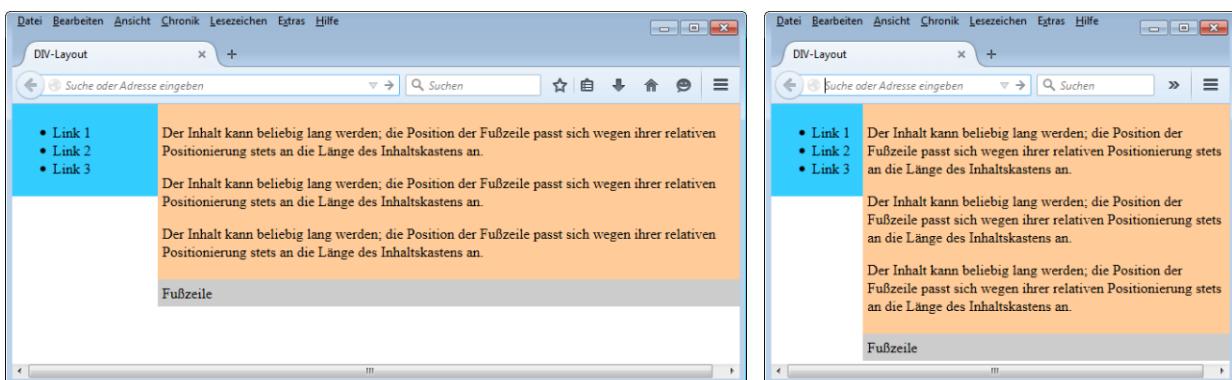
Über die Eigenschaft `margin` wird der Rand des Browsers definiert. Dieser wird von der relativen Positionierung berücksichtigt, von der absoluten hingegen nicht. Da Sie im Folgenden sowohl relative als auch absolute Positionsangaben verwenden, ist diese Definition erforderlich.

- ② Definieren Sie die Eigenschaften der Navigation. Verwenden Sie dabei eine absolute Positionierung, sodass die Navigation komplett aus der Abhängigkeit der Elemente von der Seitenposition herausgenommen wird.
- ③ Definieren Sie die Eigenschaften für Inhalt und Fußzeile. Verwenden Sie für beide Elemente eine relative Positionierung.
- ④ Erzeugen Sie jetzt die drei Container im Dokument.

Notieren Sie zuerst den absolut positionierten Container für die Navigation und erst danach die beiden relativ positionierten Container für Inhalt und Fußzeile, damit Sie eine korrekte Darstellung erhalten.



Wie die folgenden Abbildungen zeigen, handelt es sich um ein fließendes Layout, das sich einem breiteren oder schmäleren Browserfenster anpasst.



Ergebnis: Fließendes Layout

Beispiel: Textspalten mit verschachtelten div-Containern (`kap02\textbox.html`)

div-Container lassen sich auch ineinander verschachteln. Sie können diese Technik verwenden, wenn Sie bestimmte Elemente wie etwa Textspalten oder Bilder samt Beschriftung zusammenhalten möchten. So fallen nachträgliche Layoutänderungen leichter.

Der folgende Code erzeugt eine zweispaltige Textbox:

```

<style type="text/css">
<!--
#cont {
    height:200px;
    left:50px;
    position:relative;
    padding:5%;
    top:50px;
    width:200px;
}

```

```

④ #Textspalte1, #Textspalte2 {
    height:85%;
    position:absolute;
    padding:2%;
    top:5%;
    width:38%;
}
#Textspalte1 {
    left:5%;
}
#Textspalte2 {
    right:5%;
}
-->
</style>
</head>
<body>
① <div id="cont"> Kopfzeile
③   <div id="Textspalte1">Dies ist die linke Textspalte. </div>
      <div id="Textspalte2">Dies ist die rechte Textspalte.</div>
</div>
</body>
</html>

```



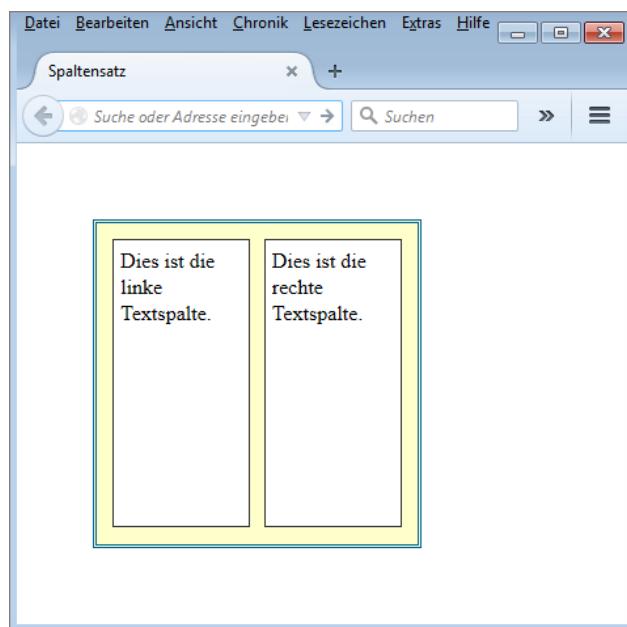
Im abgedruckten Code fehlen die Formatierungseinstellungen für Farben, Rahmeneinstellungen usw. Die Beispieldatei *textbox.html* enthält den vollständigen Code.

Erzeugen Sie zunächst einen div-Container ① mit relativer Positionierung ②.

Platzieren Sie innerhalb dieses Containers einen oder mehrere weitere div-Container ③ mit absoluter Positionierung ④.

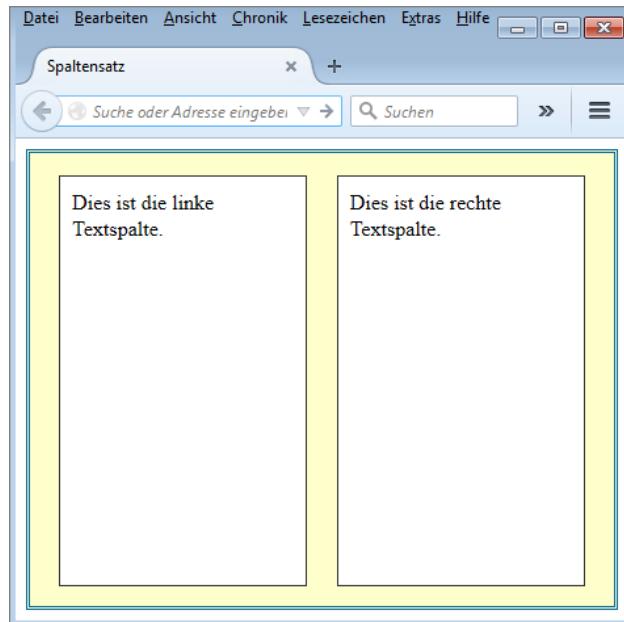
Die Positionierung der eingefügten Elemente wird nicht mehr vom Koordinatensystem der Seite, sondern ab der Position des übergeordneten Elements gemessen.

Die beiden inneren Textcontainer sind ausschließlich mit Prozentangaben formatiert. Dadurch lässt sich nicht nur die Position, sondern auch die Größe des äußeren Containers beliebig ändern - die inneren Container werden entsprechend angepasst:



Relative Positionierung ab der tatsächlichen Platzierung des div-Containers

```
#cont {
    background-color:#FFFFCC;
    border:3px double #006699;
    height:300px;
    left:0px;
    position:relative;
    padding:5%;
    top:0px;
    width:400px;
}
```



Neuformatierung des äußeren Containers mit automatischer Anpassung der inneren Container

Beispiel: Ein Layout im Browserfenster zentrieren (*kap02\zentriert.html*)

Mit der absoluten Positionierung lässt sich ein div-Container zentriert auf die Seite setzen. Unabhängig davon, wie klein oder groß das Browserfenster ist, die zentrierte Ausrichtung wird immer beibehalten.

Innerhalb dieses div-Containers können Sie dann beliebige Elemente, auch das gesamte Seitenlayout, unterbringen. Das Ergebnis ist ein vollständig zentriertes Seitenlayout.

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<style type="text/css">
<!--
#zentrierterInhalt {
    height:300px;
    left:50%;

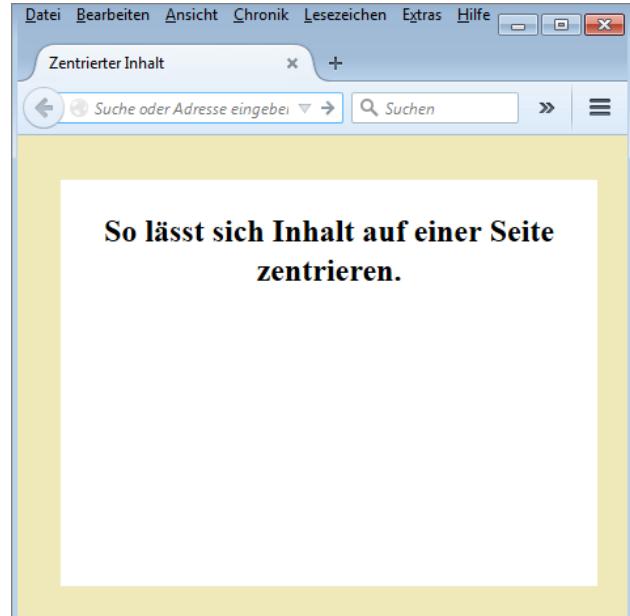
①     margin-left:-200px;
②     margin-top:-150px;
③     padding:5px;
    position:absolute;
    top:50%;
    width:400px;
}

body {
    margin:50px;
}
-->
</style>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Zentrierter Inhalt</title>
</head>
```

```
<body>
<div id="zentrierterInhalt">
    <h2 align="center">So lässt sich Inhalt auf einer Seite zentrieren. </h2>
</div>
</body>
</html>
```

Dazu müssen Sie lediglich die linke obere Ecke des div-Bereichs im Mittelpunkt der Bildschirmdarstellung anzeigen ① und dann um die Hälfte der Breite des div-Bereichs nach links ② und die Hälfte der Höhe des Bereichs nach oben ③ setzen.

 Im abgebildeten Code fehlen die Formatierungseinstellungen für Farben, Rahmeneinstellungen usw. Die Beispieldatei *zentriert.html* enthält den vollständigen Code.



Inhalt im Browserfenster zentrieren

2.7 Übungen

Übung 1: Fragen zu div-Containern und semantischen HTML-Elementen

Übungsdatei: --

Ergebnisdatei: uebung1.pdf

1. Nennen Sie die Vorteile der Layoutgestaltung mit div-Containern und semantischen HTML-Elementen gegenüber der Layoutgestaltung mit Tabellen.
2. Nennen Sie die Unterschiede zwischen einer statischen, relativen, absoluten und fixierten Positionierung.
3. Welches HTML5-Element können Sie verwenden, um eine Fußzeile zu definieren? Welches Tag können Sie alternativ verwenden, um eine möglichst breite Browserunterstützung zu erzielen?

Übung 2: Bereiche positionieren

Übungsdatei: --

Ergebnisdateien: uebung2.html, buch1k.jpg, zeitung.css

1. CSS ermöglicht ein fast perfektes Zeitungslayout. Versuchen Sie, die nachfolgende Abbildung mithilfe der gelernten Möglichkeiten der Stylesheets nachzubilden. Es geht nur um die Platzierung der Textcontainer; die Textgestaltung können Sie nach Belieben vornehmen.



In drei Spalten angeordneter Text, dargestellt in Opera

3 Elemente mit CSS anordnen

In diesem Kapitel erfahren Sie

- ✓ wie Sie den Überlauf von Elementen festlegen
- ✓ wie Sie Elemente unsichtbar machen
- ✓ wie Sie Ebenen anlegen und positionieren

Voraussetzungen

- ✓ Stylesheet-fähiger Browser

3.1 Weitergehende Möglichkeiten zur Anordnung von Elementen

In Kapitel 2 haben Sie bereits erfahren, wie Sie ein zeitgemäßes Webseitenlayout mithilfe von div-Containern und semantischen HTML-Elementen realisieren. Aber auch alle anderen HTML-Elemente können Sie positionieren. Außerdem können Sie HTML-Elemente ausblenden und in Ebenen übereinanderschichten. Auf diese Möglichkeiten wird in diesem Kapitel eingegangen.

Überlauf festlegen

Mit der Eigenschaft `overflow` (überfließen) können Sie festlegen, wie der Inhalt eines Elements dargestellt werden soll, wenn die Größe des Elements nicht ausreicht.

`overflow:Wert`

Werte für `overflow`

<code>visible</code>	Der Inhalt wird immer angezeigt, auch wenn er die angegebene Höhe überschreitet (Standard).
<code>hidden</code>	Die Höhe wird festgesetzt. Überlappende Bereiche werden abgeschnitten und nicht angezeigt.
<code>auto</code>	Der Inhalt wird abgeschnitten. Wenn nötig schaltet der Browser automatisch eine Bildlaufleiste zum Anzeigen des Inhalts ein. Kann der Inhalt innerhalb der verfügbaren Breite umbrechen, wird dabei eine vertikale Bildlaufleiste dargestellt. Ist dies nicht möglich, wird eine horizontale Bildlaufleiste angezeigt.
<code>scroll</code>	Lange Inhalte werden abgeschnitten. Der Browser zeigt immer eine vertikale und eine horizontale Bildlaufleiste an, auch wenn der Inhalt kürzer bzw. schmäler ist.

Beispiel

```
<p style="width:10cm, height:75px; overflow:auto;">Text</p>
h1 { width:50%; height:20pt; overflow:scroll; }
```

Beispiel: Overflow ändern (kap03\overflow.html)

Drei Absätze (① bis ③) erhalten bestimmte Breiten und Höhen. Mit Hilfe der Eigenschaft `overflow` werden Sie die möglichen Darstellungsformen verändern.

```

<html>
<head>
<meta charset="utf-8">
<title>width, height, overflow</title>
<style type="text/css">
    body { font:small Verdana,Arial,sans-serif; color:#000080; }
    .eins { background-color:#BCCBFE; width:150px; height:100px;
            overflow:auto; }
    .zwei { background-color:#FF0; width:10cm; height:80px;
            overflow:scroll; }
    .drei { background-color:#BCFEB; width:150px; height:50px;
            overflow:hidden; }
</style>
</head>
<body>
    <h2>Höhe und Breite und die Eigenschaft <span style="font-style:italic;">
        overflow</span></h2>
    <p class="eins">overflow:auto; Dies ist ein Absatz von 150 Pixel Breite und
        100 Pixel Höhe. Die Höhe ist zu klein, darum schaltet der Browser die
        Bildlaufleiste ein.</p>
    <p class="zwei">overflow:scroll; Dies ist ein Absatz von 10 Zentimeter Breite
        und 80 Pixel Höhe. Die Bildlaufleiste wird für das Element trotzdem
        eingeblendet.</p>
    <p class="drei">overflow:hidden; Dies ist ein Absatz von 150 Pixel Breite und
        50 Pixel Höhe. Die Höhe ist zu klein, der Text wird abgeschnitten.</p>
</body>
</html>
```

Elemente unsichtbar machen

Mit Cascading Style Sheets können Sie einzelne Elemente Ihrer Webseite unsichtbar setzen. Unsichtbar bedeutet, dass die Elemente unsichtbar werden, jedoch der Platz trotzdem eingenommen wird.

Die Angabe `visibility` (Sichtbarkeit) lässt Elemente verschwinden, macht sie sichtbar oder lässt sie die Sichtbarkeit vom übergeordneten Element erben.

<code>visibility:Wert</code>

Werte für `visibility`

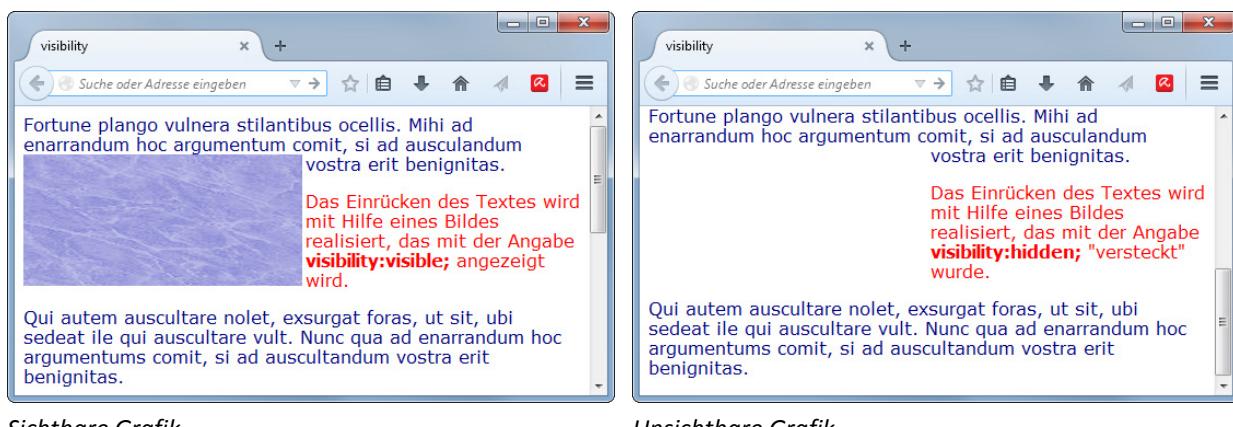
<code>visible</code>	Das Element ist sichtbar (Standard).
<code>hidden</code>	Das Element verschwindet, benötigt aber trotzdem den entsprechenden Platz auf der Webseite.
<code>inherit</code>	Die Sichtbarkeit wird vom übergeordneten Element vererbt.

Beispiel: Grafik unsichtbar machen (*kap03\visibility.html*)

```
<p style="visibility:visible">Sichtbarer
  <span style="visibility:inherit"> Text</span>
</p>
<span style="visibility:hidden"></span>
```

Mit dem Wert `visible` werden die Elemente innerhalb des Tags `<p>` angezeigt. Mit `inherit` legen Sie fest, dass der Text immer den Wert des übergeordneten Elements annehmen soll. Wenn Sie also den Wert von `visible` auf `hidden` ändern, wird auch das Wort `Text` verschwinden.

Im zweiten Teil des Beispiels wird eine Grafik unsichtbar gesetzt. Das Ergebnis können Sie in den beiden nachfolgenden Abbildungen sehen.



Teile eines Elementes anzeigen

Sie können einen Anzeigebereich bestimmen, der nur einen Ausschnitt eines Elements anzeigt, unabhängig von der eigentlichen Größe des Elements. Wenn das Element größer ist als der definierte Anzeigebereich, wird dessen Inhalt an den entsprechenden Seiten abgeschnitten.

clip:rect()	Mit <code>clip:rect()</code> können Sie einen rechteckigen Ausschnitt für die sichtbare Anzeige definieren. In den runden Klammern werden vier numerische Werte oder das Schlüsselwort <code>auto</code> zur Bestimmung des Ausschnitts festgelegt. Dabei gelten die folgenden Abstände: <code>clip:rect(oben, Breite von links, Höhe von oben, links)</code> Die Längenangaben können Sie in den üblichen CSS-Einheiten angeben. Das Schlüsselwort <code>auto</code> bedeutet: keine Angabe zu dem betreffenden Wert. Das Element soll an der entsprechenden Seite vollständig angezeigt, also nicht beschnitten werden. Derzeit können Sie nur ein Rechteck für den Ausschnitt definieren. In Zukunft sollen auch Polygone, Kreise bzw. Ellipsen möglich sein.
--------------------	---

Beispiel: Grafik zuschneiden (*kap03\clip.html*)

```

p { position:absolute; clip:rect(10%, 90%, auto, 10%); }
```

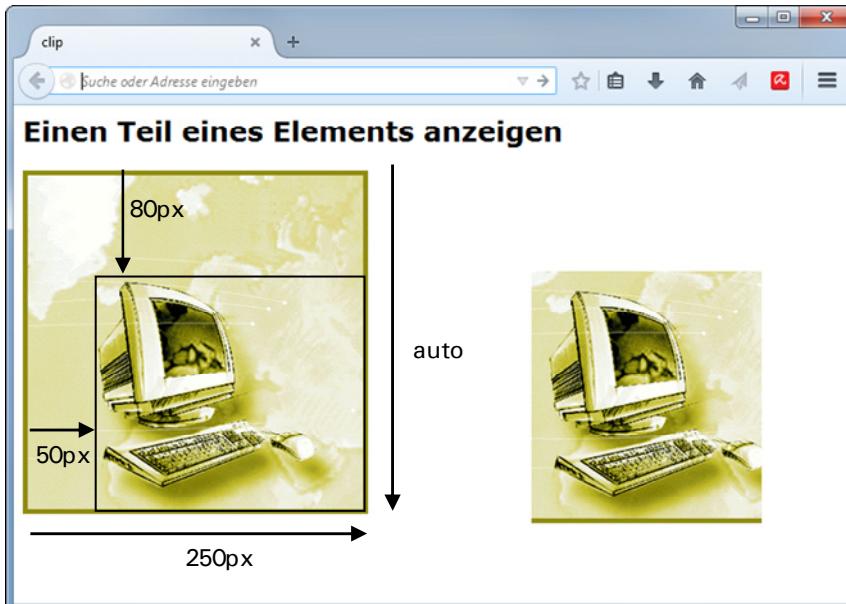
Diese Eigenschaft kann nur angewendet werden, wenn das Element über die Eigenschaften `position: absolute` oder `position:fixed` positioniert ist. Ansonsten wird der verkleinerte Anzeigebereich nicht umgesetzt.

Die Angabe der einzelnen Werte ist etwas gewöhnungsbedürftig, da sie in einer anderen Reihenfolge als sonst üblich angegeben werden. Der erste Wert entspricht der oberen Grenze des Elements. Der zweite Wert bezieht sich auf die rechte, der dritte auf die untere und der letzte Wert auf die linke Grenze. Die Werte beziehen sich dabei immer auf die linke obere Ecke des gewählten Elements.

Als Faustregel sollten Sie sich merken, dass der zweite Wert immer größer als der vierte Wert und der dritte Wert immer größer als der erste Wert sein müssen, damit ein Teil des Elements angezeigt werden kann.



In der nachfolgenden Abbildung wurde eine Grafik mit einer Höhe und Breite von jeweils 300 Pixeln normal in eine Webseite eingefügt. Auf der rechten Seite wird die Grafik auf einen sichtbaren Teil reduziert.



Gegenüberstellung von normalem und abgeschnittenem Bild (kap06\clip.html)

In diesem Beispiel wurde die Grafik mit der Angabe von `clip: rect (80px, 250px, auto, 50px)` beschnitten. Dies bedeutet:

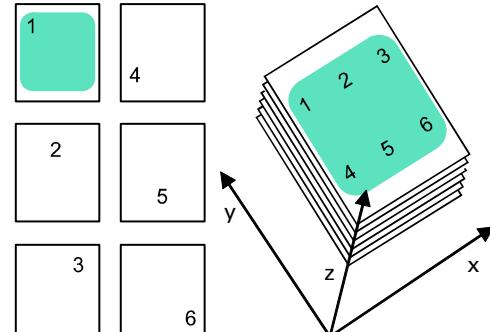
- ▶ Vom oberen Rand werden 80 Pixel abgeschnitten.
- ▶ Vom rechten Rand werden 50 Pixel abgeschnitten (Originalgröße 300 Pixel - 250 Pixel = 50 Pixel).
- ▶ Vom unteren Rand wird nichts abgeschnitten (`auto`).
- ▶ Vom linken Rand werden 50 Pixel abgeschnitten.

3.2 Ebenen anlegen

Stellen Sie sich Ebenen wie durchsichtige Folien vor, die übereinandergelegt werden.

Die erste (unterste) Folie ist die 1. Ebene, die zweite, darüber liegende Folie ist die 2. Ebene usw. Da die Ebenen durchsichtig sind, sind die Elemente der einzelnen Ebenen sichtbar.

Dabei ist auch der Effekt möglich, dass ein Element einer Ebene ein Element einer anderen Ebene überlagert. Bisher war es in HTML nicht möglich, einzelne Elemente übereinander zu platzieren.



Aufbau von Ebenen

In CSS werden die Ebenen automatisch überlappend angelegt, wenn Sie Elemente absolut oder fixiert positionieren. Dabei wird jede neue Ebene automatisch über die bereits vorhandenen Ebenen gelegt. Diese Reihenfolge können Sie mit der folgenden CSS-Eigenschaft beeinflussen.

z-index: Werte: positive und negative Ganzzahlen	Wenn Sie mehrere Elemente positionieren und diese einander überlappen sollen, benutzen Sie z-index, um anzugeben, wie die Elemente angeordnet werden sollen. Der z-index wurde aus dem Koordinatensystem übernommen: x und y stellen die Ausdehnungen in der Breite und Höhe dar, z gibt die Tiefe eines Elements an. Der Wert der z-Koordinate, den Sie den einzelnen Bereichen zuweisen, muss eine Ganzzahl sein. Dezimalzahlen sind nicht möglich. Elemente mit höheren Werten überdecken die Bereiche von Elementen mit niedrigeren Werten.
--	--

Beispiel

```
<p style="position:absolute; left:10px; top:0px; z-index:10;">Text</p>
h1 { position:relative; left:20px; top:-5px; z-index:-5; }
```

Beispiel: Z-Index ändern ([kap03\z-index.html](#))

Verschiedene Blockelemente werden so angeordnet, dass sie sich teilweise überlappen. Die Reihenfolge der mit div festgelegten Elemente weicht dabei von der automatischen Zuweisung der Ebenen ab.

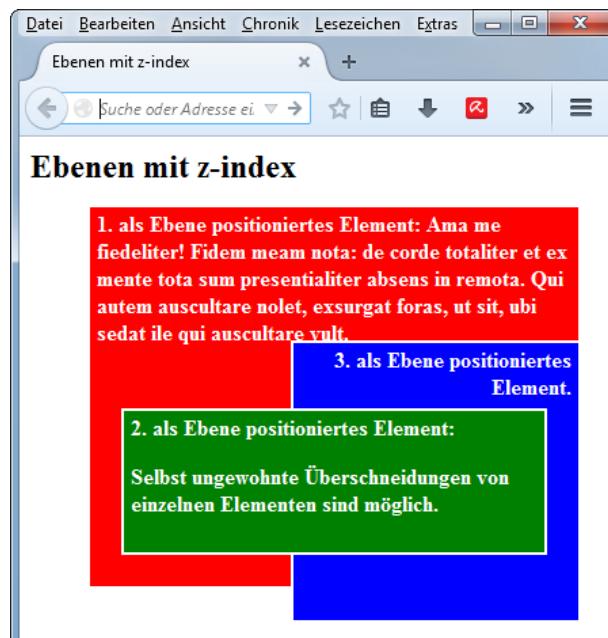
	<pre><html> <head> <meta charset="utf-8"> <title>Ebenen mit z-index</title> <style type="text/css"></pre>
①	<pre> div { position:absolute; border:2px solid white; color:white; font-weight:bold; padding:2px 5px; }</pre>
②	<pre> .nr1 { top:50px; left:50px; width:350px; height:275px; background-color:red; z-index:10; }</pre>
③	<pre> .nr2 { top:200px; left:75px; width:300px; height:100px; background-color:green; z-index:30; }</pre>

```

④ .nr3 { top:150px; left:200px; width:200px; height:200px;
          background-color:blue; text-align:right; z-index:20; }
      </style>
</head>
<body>
  <h2>Ebenen mit z-index</h2>
⑤ <div class="nr1">1. als Ebene positioniertes Element: Ama me fiedeliter!
    Fidem meam nota: de corde totaliter et ex mente tota sum ...</div>
  <div class="nr2">2. als Ebene positioniertes Element:<p>Selbst ungewohnte
    Überschneidungen von einzelnen Elementen sind möglich.</p></div>
  <div class="nr3">3. als Ebene positioniertes Element.</div>
</body>
</html>

```

- ① Zu Beginn legen Sie fest, dass alle div-Elemente absolut zu positionieren sind. Zur besseren Hervorhebung erhalten alle div-Elemente einen weißen, 2 Pixel breiten Rand. Der Text wird ebenfalls weiß und fett formatiert. Zusätzlich besitzt er einen geringen Innenabstand zum Rahmen.
- ② Mit dem Klassenselektor nr1 legen Sie die Position, die Höhe und die Breite des ersten Elements fest. Die Hintergrundfarbe ist Rot. Das Element, dem diese Klasse zugewiesen wird, erhält den Ebenenwert 10.
- ③ Der Selektor nr2 formatiert das Element entsprechend den Positionswerten, setzt einen grünen Hintergrund und setzt das Element auf die Ebene 30. Somit liegt dieses Element über dem Element mit der Klasse nr1.
- ④ Die letzte Selektordefinition erwirkt einen blauen Hintergrund. Der Text wird rechts ausgerichtet. Mit dem z-index-Wert von 20 liegt das formatierte Element zwischen den beiden Elementen mit den Klassen nr1 und nr2. Es überlappt dann zwar das Element mit der Klasse nr1, wird jedoch selbst teilweise vom Element mit der Klasse nr2 überdeckt.
- ⑤ Es folgt die Zuweisung der einzelnen Selektoren und somit der Formate.



Das Ergebnis der sich überlappenden Elemente

3.3 Responsives Webdesign

Heutzutage greift ein großer Anteil der Nutzer über mobile Geräte wie etwa Smartphones und Tablets auf Internetseiten zu. Aus diesem Grund sollten moderne Webseiten immer auch für solche Ausgabemedien optimiert sein.

Die Suchmaschine Google bewertet Webseiten, die für mobile Geräte optimiert sind, mittlerweile besser und zeigt diese auf der SERP (Suchergebnisseite) höher an als Seiten ohne eine solche Optimierung.



Man spricht in diesem Zusammenhang von „responsivem Webdesign“ oder RWD: Die Webseite antwortet („responds“) auf die Display-Größe des jeweils verwendeten Ausgabegeräts und passt sich dieser flexibel an.

Zum Beispiel könnten die Inhalte auf einem großen PC-Bildschirm mehrspaltig dargestellt werden, auf dem kleinen Display eines Smartphones einspaltig und untereinander.

@media-Regel

Ein wichtiges Hilfsmittel für responsives Webdesign ist die @media-Regel von CSS3. Mit dieser können Sie innerhalb eines Stylesheets für bestimmte Elemente mehrere Formatangaben definieren, die je nach Ausgabegerät zum Tragen kommen.

<pre><code>@media screen { body {color: #006} }</code></pre>	<p>Auf @media folgt die Angabe des Medientyps. Sie können hier auch mehrere Medientypen angeben, getrennt durch Kommas.</p> <p>Innerhalb der darauffolgenden, äußeren geschweiften Klammern stehen in gewohnter Notierungsweise die CSS-Anweisungen, die für den bzw. die angegebenen Medientyp(en) gelten sollen.</p> <p>Mögliche Medientypen:</p> <ul style="list-style-type: none"> ✓ screen: Computerbildschirme (und tragbare Geräte wie Tablets und Smartphones) ✓ tty: Terminals mit grober Auflösung ✓ tv: Fernseher ✓ projection: Projektoren ✓ handheld: Tragbare Geräte mit geringer Bandbreite ✓ print: Drucker ✓ braille: Braillezeile (Ausgabegerät für Blindenschrift) ✓ embossed: Brailledrucker ✓ speech: Sprachausgabe ✓ all: alle Medientypen
--	--

Wichtig sind die folgenden CSS-Eigenschaften, da Sie damit die Breite des Browserfensters bzw. des Viewports abfragen können:

<code>min-width:</code>	Fragt die mindestens nutzbare Breite des Viewports ab. Beim PC-Bildschirm ist das der innere Bereich des Browserfensters (bei den meisten Browsern inklusive Rollbalken). Beim Smartphone ist es die dargestellte Größe, die deutlich höher liegt als die wirkliche Größe des Displays.
<code>min-device-width:</code>	Fragt die mindestens nutzbare Breite des Ausgabegeräts ab.
<code>max-width:</code>	Fragt die höchstens nutzbare Breite des Viewports ab.
<code>max-device-width:</code>	Fragt die höchstens nutzbare Breite des Ausgabegeräts ab.

Beispiel: Einfache Media Queries (*kap03\media-queries.html*)

Ein Beispiel zeigt Ihnen, wie Media Queries in ihrer einfachsten Form funktionieren. Je nach Größe des Browserfensters soll sich die Randbreite der Webseite ändern.

- ▶ Legen Sie ein neues Dokument an, und speichern Sie es unter dem Namen *media-queries.html*.
- ▶ Geben Sie einen beliebigen Text ein.
- ▶ Geben Sie vor dem *</head>*-Tag den folgenden CSS-Code ein:

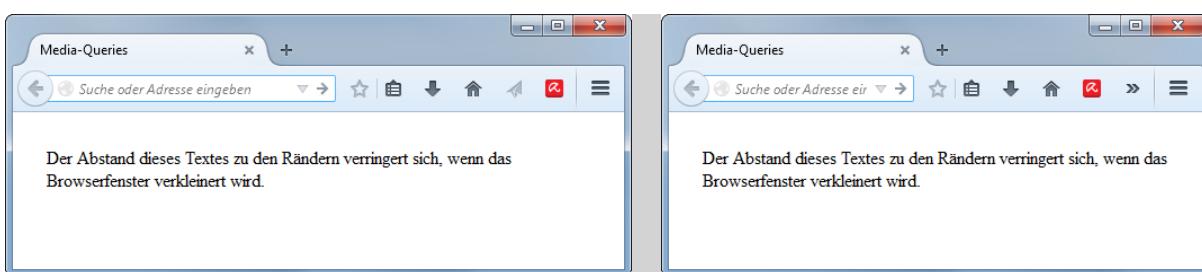
```

① <style type="text/css">
    body {margin: 70px;}
② @media screen and (max-width: 640px) {
    body {
        margin: 30px;
    }
}
③ @media screen and (max-width: 320px) {
    body {
        margin: 5px;
    }
}
</style>

```

- ① Das body-Element hat die Eigenschaft margin: 70px erhalten. Damit erhält die Webseite allseits einen 70 Pixel breiten Rand.
- ② Um zu verhindern, dass bei einem kleinen Bildschirm (maximal 640 Pixel) der Rand zu breit wirkt, wird dieser mit einer Media Query auf 30 Pixel gesetzt.
- ③ Ist das Ausgabe-Display wie etwa bei einem Smartphone noch kleiner (maximal 320 Pixel breit), wird der Rahmen auf 5 Pixel gesetzt.

Wird das Browserfenster kleiner gezogen, verringert sich bei zwei Breiten (640 Pixel und 320 Pixel) die Randbreite:



Meta-Element `viewport`

Bei Smartphones wird die gesamte Webseite standardmäßig so stark kleiner gezoomt, dass sie komplett auf das Display des Geräts passt. Dies ergibt den **Viewport** des Smartphones, der mit width gemessen wird. Er kann etwa bei 980 Pixel liegen (iPhone), obwohl das Smartphone-Display eigentlich nur 320 Pixel breit ist (device-width).

Definieren Sie auf Ihrer Webseite beispielsweise ein Element mit festen Pixelabmessungen, wird es auf dem Smartphone-Display unter Umständen sehr klein dargestellt.

Dies können Sie mit der Angabe width=device-width verhindern. Sie definieren diese im head-Bereich Ihrer Seite:

```
<meta name="viewport" content="width=device-width" />
```

Auf Desktop-Browsern hat diese Metaangabe keine Auswirkungen.



Weitere Meta-Angaben:

user-scalable=no	Das Zoomen wird deaktiviert.
initial-scale=1.0	Die Inhalte werden 1:1 dargestellt. Mit dem Wert 2.0 würden Sie beispielsweise eine zweifache Vergrößerung erzielen.
maximum-scale=2.0	Die Webseite kann maximal auf das Zweifache gezoomt werden.
minimum-scale=2.0	Die Webseite kann maximal um die Hälfte verkleinert werden.



Damit ein Web-Layout responsiv wird, sollten alle Breitenangaben in Prozent definiert sein. Nur dann können sich alle Elemente von der Breite her stets an das Ausgabegerät anpassen.

Von zweispaltig zu einspaltig wechseln

Zweispaltige Layouts sollen meist einspaltig dargestellt werden, wenn der Viewport eine bestimmte Breite unterschreitet.

Beispiel: Media Query mit neuer Layoutanordnung (*kap03\spalten-responsiv.html*)

- ▶ Öffnen Sie die Beispieldatei *spalten.html*.

Der Code erzeugt ein einfaches, zweispaltiges Layout.



Um den nachfolgend abgedruckten Code knapp zu halten, wurden die Formatierungen der Hintergründe usw. weggelassen.

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Media-Query</title>
<style type="text/css">
<!--
#navigation {
    position: absolute;
    width: 20%;
}
#inhalt {
    position: relative;
    width: 80%;
    left: 20%;
}
#fusszeile {
    position: relative;
    width: 80%;
    left: 20%;
}
-->
</style>
</head>

<body>
<nav id="navigation">
```

```
<ul>
    <li>Link 1 </li>
    <li>Link 2</li>
    <li>Link 3</li>
</ul>
</nav>
<div id="inhalt">
    <p>...</p>
</div>
<footer id="fusszeile"> Fußzeile </footer>
</body>
</html>
```

Dieses Layout ist nicht responsiv. Wird das Browserfenster sehr schmal gezogen, sieht die Webseite nicht mehr gut aus und wirkt unübersichtlich.

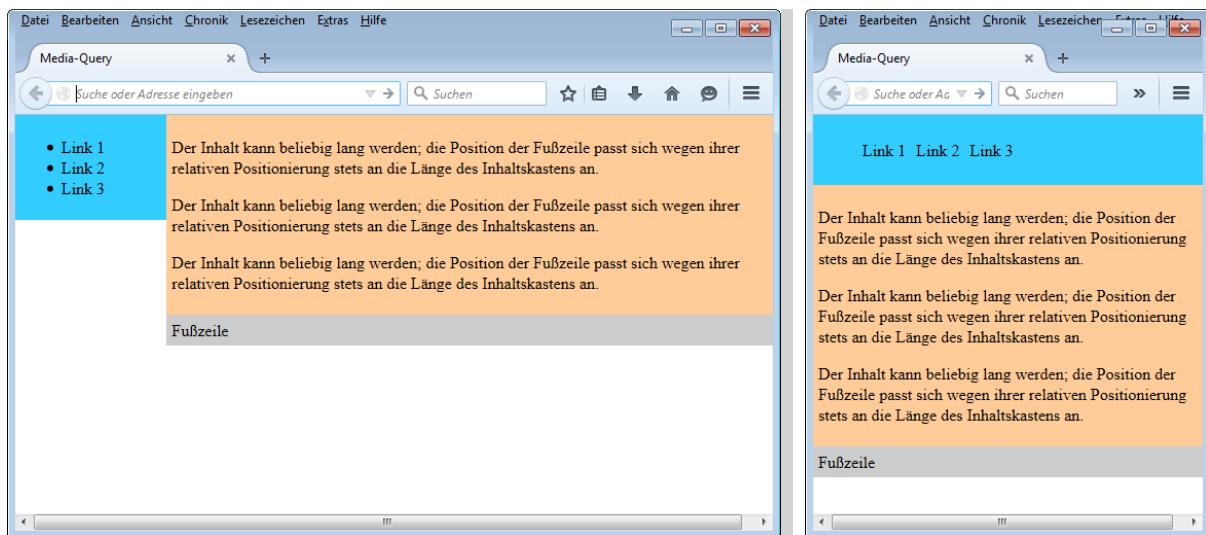
Deshalb soll das Layout einspaltig dargestellt werden, sobald sich die Breite des Viewports auf 450 px oder weniger verringert. Die Navigationsleiste mit den vier Links soll nicht mehr links neben dem Inhaltsbereich, sondern horizontal darüber angezeigt werden.

- Fügen Sie eine Media Query ein, um das Layout bei 450 px oder weniger Viewportbreite einspaltig darzustellen:

```
① @media screen and (max-width: 450px) {
    ② #inhalt, #navigation, #fusszeile {
        width: 100%;
        position: inherit;
    }
    ③ #navigation li {
        list-style: none;
        display: inline-block;
        ④ padding: 3px;
    }
}
```

- ① Geben Sie `#inhalt`, `#navigation` und `#fusszeile` jeweils eine Breite von 100%.
- ② Die relative Positionierung entfernen Sie mit `inherit`.
- ③ Entfernen Sie die Bullets vor den einzelnen Links.
- ④ Zeigen Sie die Liste als inline-Block mit Abständen von 3px an.

Ab 450 Pixel wird das zweispaltige Layout einspaltig dargestellt:



Desktop first und Mobile first

In Abschnitt 3.3 haben Sie erfahren, wie Sie – ausgehend von einem zweispaltigen Layout – eine Version für Ausgabegeräte mit kleinflächigem Display erzeugen. Diese Herangehensweise an das responsive Layout wird **Desktop first** genannt.

In der Praxis bietet es sich heutzutage an, stets zuerst das mobile Layout zu entwickeln und anschließend eine Media Query hinzuzufügen, die das Layout für die Darstellung auf einem Desktop-Computer optimiert. Diese Strategie heißt **Mobile first**.

Fertige Lösungen für responsives Weblayout



Je komplexer Ihre Layouts, desto aufwendiger wird die Erstellung responsiver Lösungen: Unter anderem muss möglicherweise auch die Größe von Grafiken flexibel angepasst werden, bestimmte Elemente sollen bei der mobilen Version ganz weggelassen werden usw.

Hilfreich sind deshalb vorgefertigte Lösungen, wie sie etwa im Framework Twitter Bootstrap enthalten sind. Sie erhalten hier ein zwölfspaltiges Rasterlayout, das Sie nach Ihren Wünschen ändern können und das sich automatisch verschiedenen Auflösungen anpasst.

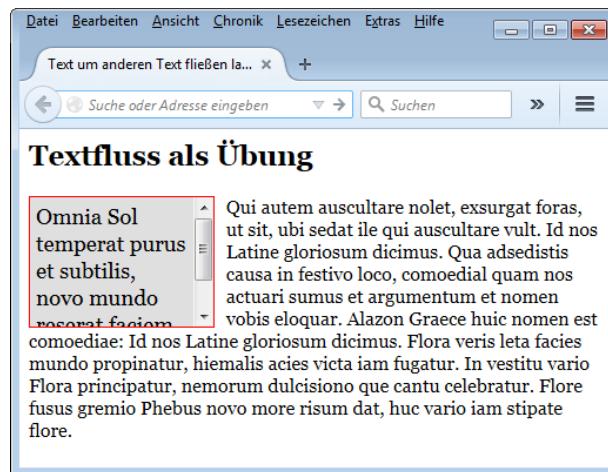
3.4 Übungen

Übung 1: Text um anderen Text fließen lassen

Übungsdatei: --

Ergebnisdatei: uebung1.html

1. Erstellen Sie die nebenstehende Webseite, bei der der linke vom rechten Text umflossen wird. Setzen Sie den linken Text zur Hervorhebung in einen separaten Rahmen. Die Breite beträgt 30 % des Browserfensters, und die Höhe ist mit 100 Pixeln vorgegeben. Ist der linke Text länger, dann soll für das Element die Bildlaufleiste eingeblendet werden.



Textumfluss

Übung 2: Responsives Layout - „Mobile first“

Übungsdatei: --

Ergebnisdatei: uebung2.html

1. Arbeiten Sie das in Abschnitt 3.3 gezeigte Beispiel so um, dass es zur Mobile first-Strategie passt.

4 Sound und Videos einbinden

In diesem Kapitel erfahren Sie

- ✓ wie Sie Sound und Videos einbinden
- ✓ welche Multimediaformate es gibt

4.1 Multimedia im Web

Im Web werden in immer stärkerem Maß multimediale Technologien wie Videos oder Musik eingesetzt, um die Attraktivität der Seiten zu erhöhen.

Vor HTML5: Plug-ins

Bis HTML4 ist Hypertext nicht multimediafähig. Das bedeutet, dass der Internetbrowser Unterstützung benötigt, um Sound- oder Videodateien anzeigen zu können. Dies kann mit Plug-ins realisiert werden. **Plug-ins** sind an den Browser angekoppelte oder von ihm abhängige Zusatzdateien mit den benötigten Programmfunctionen. Sie ermöglichen dem Browser, fremde Dateiformate auf der Webseite anzuzeigen, auch wenn der Browser diese Dateiformate nicht direkt unterstützt.

Voraussetzung hierfür ist, dass der Anwender das entsprechende Plug-in auf seinem Computer installiert hat.



Die wichtigsten Plug-ins sind das Animationsformat Adobe Flash, der Dokumentanzeiger Acrobat Reader für die Anzeige von PDF-Dateien, die Audio-Plug-ins für die Formate MP3, MIDI, WAVE, Real Audio sowie die Plug-ins zum Anschauen von Videos in Formaten wie etwa AVI, MPEG und QuickTime.

Viele Plug-ins, zum Beispiel für Sound, digitales Video sowie der Flash-Player, werden bereits mit den Browsern ausgeliefert, andere muss der Benutzer nachträglich von der Website des Herstellers herunterladen und installieren. Sie sind oft kostenlos.

Plug-ins einbinden

Um Mediendateien per Plug-ins in die Webseite einzubinden, verwenden Sie das `object`-Element. Sobald Sie eine Webseite mit einer `object`-Anweisung aufrufen, ermittelt der Browser im Hintergrund zunächst, welches Plug-in für die Mediendateien installiert ist, und lädt dieses anschließend.

Das Attribut `data` des `object`-Elements enthält die Referenz zu der entsprechenden Objektdatei und muss in Anführungszeichen angegeben werden. Dabei spielt es keine Rolle, welches Format das Objekt hat. Dieses legen Sie über das Attribut `type` und das entsprechende MIME-Format fest. Die Attribute `width` und `height` bestimmen die Größe der anzugezeigenden Datei im Browser. Diese Attribute müssen Sie angeben, da ansonsten das Objekt nicht angezeigt werden kann. Mit der Angabe `standby` können Sie einen alternativen Text angeben, der angezeigt wird, bis das Objekt vollständig geladen ist.

Für einige eingebundene Elemente werden über das optionale Tag `param` ein oder mehrere Parameter übergeben. Dabei werden der Name (`name`) und der Wert (`value`) des gewünschten Parameters angegeben:

```
<object data="clock.avi" type="video/x-msvideo" width="100" height="100"  
       standby="Lade AVI"></object>  
<object data="universal.html" type="text/html" width="400"  
       height="200"></object>
```

In der ersten Zeile binden Sie ein Video in die Webseite ein, das in einer Größe von 100 x 100 Pixeln dargestellt wird. Auch Webseiten lassen sich damit einbinden, wie die zweite Zeile zeigt. Die genaue Größe legen Sie über die beiden Attribute `width` und `height` fest.

HTML5: Sound und Videos direkt einbinden

In HTML5 können Sie multimediale Elemente wie Audio- und Videodateien einbinden, ohne dazu wie bisher Plug-ins wie QuickTime oder Flash zu benötigen.

Sie verwenden dazu die neuen HTML5-Elemente `<audio>` und `<video>`. Dies hat gegenüber der zuvor erläuterten Lösung mit der Verwendung des `object`-Elements eine Reihe von Vorteilen:

Vereinfachte Einbindung	Die meisten Plug-ins sind kostenlos, ihre korrekte Einbindung bedeutet jedoch einen gewissen Entwicklungsaufwand. Mit den HTML5-Elementen <code>audio</code> und <code>video</code> ist dieser Vorgang standardisiert und deutlich vereinfacht worden und es sind keine Plug-ins notwendig.
Keine Probleme mit Nutzungsrechten und Werbung	Zwar ist es kein Problem, ein Video auf YouTube oder Vimeo zu laden und den Code in die eigene Website zu integrieren, jedoch gehen die Nutzungsrechte dann normalerweise an den Plattformbetreiber und es wird Werbung in die Videos eingeblendet. Durch die Elemente <code>audio</code> und <code>video</code> ist auch dieses Problem behoben.

Beachten Sie, dass die `<audio>`- und `<video>`-Elemente nur von den folgenden Browsern unterstützt werden:

- ✓ Internet Explorer ab Version 9
- ✓ Firefox ab Version 3.5
- ✓ Safari ab Version 3
- ✓ Google Chrome ab Version 3
- ✓ Opera ab Version 10.5
- ✓ iPhone ab Version 1
- ✓ Android ab Version 2



Ein zusätzliches Problem ist, dass nicht alle Multimediaformate von jedem Browser ohne zusätzliche Software abgespielt werden können. So ist beispielsweise das weitverbreitete Audioformat MP3 patentgeschützt, sodass es nicht in jeden Browser implementiert werden kann (vgl. Abschnitt 4.2).



Damit Nutzer mit älteren Browsern Ihre Multimedia-Dateien abspielen können, ist eine Lösung (Fallback) notwendig, die bei der ausführlichen Erläuterung der Elemente `audio` und `video` behandelt wird (vgl. Abschnitt 4.3).

4.2 Multimediaformate kennenlernen

Codecs

Digitale Multimedia-Dateien sind normalerweise mehrere Megabyte groß. Deshalb könnten sie ohne Kompression (auch Komprimierung genannt) nicht über das Internet gestreamt werden. Dazu werden verschiedene sogenannte **Codecs** verwendet.

Codec ist ein Kunstwort aus Coder/Decoder. Es handelt sich dabei um ein Verfahren zur Komprimierung während der Aufzeichnung und der Dekomprimierung während des Abspielens von Mediendateien. Durch die Kompression wird die Audio-/Videodatei kleiner. Dabei werden redundante Informationen sowohl zwischen den einzelnen Frames als auch in den Frames selbst entfernt.

Dadurch kommt es stets zu einem mehr oder minder ausgeprägten Daten- und damit Qualitätsverlust. Die Komprimierung ist stets ein Kompromiss zwischen möglichst geringer Dateigröße und möglichst hoher Qualität der Mediendatei.



Außerdem sind viele Codecs durch Patente geschützt, die zwar nicht den Endanwender betreffen, wohl aber Softwarehersteller, die diese Codecs nutzen möchten. Patentgeschützte Codecs bieten momentan eine deutlich höhere Kompressionsrate als patentfreie Codecs. Die Browser Mozilla und Opera setzen auf patentfreie Codecs mit geringerer Kompressionsrate. Die Browser von Microsoft, Google und Apple verwenden patentgeschützte Codecs mit besserer Kompressionsrate.



Deshalb sollten Sie Multimedia-Dateien in HTML5-Webseiten in mehreren Formaten anbieten. Im weiteren Verlauf dieses Kapitels erfahren Sie, wie das geht.

Die derzeit wichtigsten Audioformate/Codecs für das Web sind:

.mp3 MPEG-1 Audio Layer 3 (MP3)	MP3 – kurz für MPEG-1 Audio Layer 3 – komprimiert Audiodateien verlustbehaftet. Die Kompression erfolgt dabei so, dass die Wahrnehmung je nach Kompressionsstufe nicht oder nur gering beeinträchtigt wird. Die Kompressionsrate ist hoch, die resultierenden Dateien sind klein, sodass MP3 sich schnell zum wichtigsten Audioformat für das Web entwickelt hat.
.ogg OGG/Vorbis	Das Container-Format OGG eignet sich für Multimedia-Inhalte aller Art - Video, Audio und Text. Es können unterschiedliche Audio- und Video-Codecs enthalten sein. Üblicherweise wird für OGG-Audio der patentfreie Vorbis-Codec verwendet. Ähnlich wie MP3 komprimiert auch Vorbis verlustbehaftet. Der Nachteil gegenüber MP3 ist, dass die resultierenden Dateien größer sind.

Die folgende Tabelle zeigt Ihnen die derzeit wichtigsten Videoformate/Codecs für das Web:

.mp4 MPEG-4/H.264	Der ISO-Standard MPEG-4 ist aus dem QuickTime-Format von Apple hervorgegangen. Es kann unter anderem mehrere Video- und Audiospuren speichern. Für das MPEG-4-Format wird der durch Patente geschützte Video-Codec H.264 verwendet.
.ogv OGV/Theora	Der patentfreie Video-Codec Theora ist der übliche Codec für OGG-basierte Videos. Die Kompressionsrate ist weniger hoch als die des H.264-Codecs.

Multimedia-Dateien encodieren

Bevor Sie Ihre eigenen Multimedia-Dateien in HTML5 einbinden, müssen Sie sie also encodieren. Für .mp3 und .mp4 können Sie dazu beispielsweise den Adobe Media Encoder o. Ä. verwenden. Für das OGG-Format gibt es verschiedene frei verfügbare Encoder, die Sie sich aus dem Internet herunterladen können, beispielsweise Miro Video Converter.

4.3 Sound in HTML5 einbinden

Zum Einbinden von Audio in HTML5 verwenden Sie das `<audio>`-Element.

audio	Innerhalb des <code>audio</code> -Elements setzen Sie einen oder mehrere Verweise auf die Sounddatei oder mehrere alternative, in unterschiedlichen Dateiformaten gespeicherte Versionen der Sounddatei.
--------------	--

Folgende Attribute stehen für das audio-Element zur Verfügung:

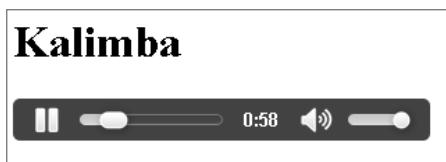
src	Die URL der abzuspielenden Audiodatei
preload	Der Browser beginnt direkt beim Laden mit dem Download der Audiodatei.
autoplay	Der Inhalt wird bereits während der Übertragung aus dem Internet im Browser des Endnutzers abgespielt.
loop	Wird dieses Attribut angegeben, wird die Audiodatei in einer Endlosschleife abgespielt.
controls	Die Steuerelemente (Abspielen/Pausieren/Position/Lautstärke) werden auf der Webseite angezeigt.

Beispiel: Audio in Webseite einbetten (*kap04\audio1.html*)

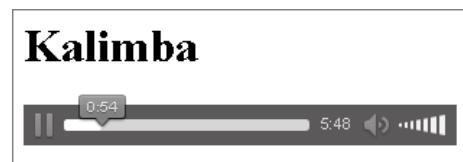
```
<body>
<h1>Kalimba </h1>
<div>
① <audio preload controls src="Kalimba.mp3">
②   <source src="Kalimba.ogg">
</audio>
</div>
</body>
```

- ① Mit dem Tagpaar `<audio> ... </audio>` binden Sie eine Audio-Ressource in die Webseite ein. Mithilfe des Attributs `src` dieses Elements geben Sie die URL der abzuspielenden Audiodatei an. Das Attribut `preload` sorgt dafür, dass der Browser direkt beim Laden mit dem Download der Audiodatei beginnt. Das Attribut `controls` sorgt dafür, dass ein Audio-Player auf der Webseite angezeigt wird. Mit diesem kann die Audiodatei abgespielt, pausiert und angehalten sowie ihre Lautstärke verändert werden.
- ② Mit dem `source`-Element innerhalb des `audio`-Elements können Sie den Namen einer alternativ abzuspielenden Audiodatei angeben, falls der Browser das Dateiformat der direkt im `audio`-Element angegebenen Datei nicht interpretieren kann. Sie können hier auch mehrere `source`-Elemente untereinander angeben.

Die Soundkonsole sieht in jedem Browser etwas anders aus.



Soundkonsole in Google Chrome



... in Firefox

Die direkt im `audio`-Element angegebene Sounddatei sollte ein Format haben, das der Browser auf jeden Fall erkennt. Zum Zeitpunkt der Erstellung des Buchs war unklar, welches Format sich durchsetzen wird. Sie können alternativ im Code z. B. zwei Formate gleichwertig nebeneinander angeben:



Beispiel: Audio in Webseite einbetten (*kap04\audio2.html*)

```
<body>
<h1>Kalimba </h1>
<div>
① <audio preload controls>
②   <source src="Kalimba.mp3" type="audio/mpeg" />
    <source src="Kalimba.ogg" type="audio/ogg" />
</audio>
</div>
</body>
```



Das Element `source` kennt zwei Attribute:

<code>type</code>	Gibt ggf. den MIME-Typ der Audiodatei an, etwa <code>audio/x-midi</code> , <code>audio/x-wav</code> usw. Sie können das <code>type</code> -Attribut auch mit der Angabe des Codecs versehen, beispielsweise: <code>type="audio/ogg; codecs=vorbis"</code>
<code>media</code>	Gibt an, für welche Medientypen die Audiodatei geeignet ist. Die Voreinstellung lautet: <code>media="all"</code>



Diese Lösung funktioniert nur in modernen Browsern, die HTML5 unterstützen. Möchten Sie Ihre Multimedia-Dateien für ein möglichst breites Publikum zugänglich machen, können Sie auf JavaScript-Code zurückgreifen, der eine Lösung für ältere Browser bietet – diese spielen statt des unbekannten `audio`-Elements eine Flash-Datei ab. Ein Beispiel finden Sie in der Datei *audio-fuer-alle.html*.

4.4 Videos einbinden

Zum Einbinden eines Videos in HTML5 verwenden Sie das `video`-Element.

<code>video</code>	Innerhalb des <code>video</code> -Elements setzen Sie einen oder mehrere Verweise auf die Videodatei oder mehrere alternative, in unterschiedlichen Dateiformaten gespeicherte Versionen der Videodatei.
--------------------	--

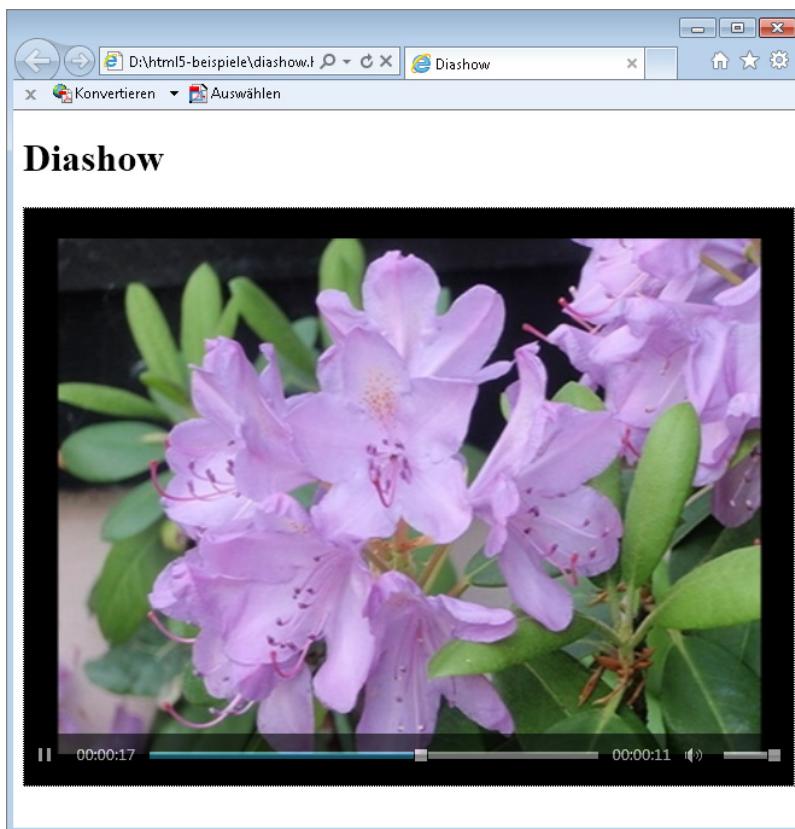
Folgende Attribute stehen für das `video`-Element zur Verfügung:

<code>src</code>	Die URL der abzuspielenden Videodatei
<code>preload</code>	Der Browser beginnt direkt beim Laden mit dem Download der Videodatei.
<code>autoplay</code>	Der Inhalt wird bereits während der Übertragung aus dem Internet im Browser des Endnutzers abgespielt.
<code>loop</code>	Wird dieses Attribut angegeben, wird die Videodatei in einer Endlosschleife abgespielt.
<code>controls</code>	Eine Abspielsteuerung für das Video wird auf der Webseite angezeigt.
<code>poster</code>	URL einer Grafik, die angezeigt wird, falls die Videodatei nicht abgespielt werden kann
<code>width</code>	Die Breite der Videoanzeige in Pixeln
<code>height</code>	Die Höhe der Videoanzeige in Pixeln

Beispiel: Video in Webseite einbetten (*kap04\video.html*)

```
<h1>Diashow</h1>
① <video src="diashow.mp4" width="640" height="480"
       controls preload poster="diashow.jpg">
    <source src="diashow.ogv" type="video/ogg"></source>
</video>
```

- ① Mit dem Tagpaar `<video> ... </video>` binden Sie eine Video-Ressource in die Webseite ein. Mithilfe des Attributs `src` geben Sie die URL der abzuspielenden Videodatei an. Das Attribut `preload` sorgt dafür, dass der Browser direkt beim Laden der Seite mit dem Download der Videodatei beginnt. Das Attribut `controls` sorgt dafür, dass ein Video-Player auf der Webseite angezeigt wird. Mit diesem kann die Videodatei abgespielt, pausiert und angehalten sowie ihre Lautstärke verändert werden. Das Attribut `poster` verweist auf eine Grafik, die angezeigt wird, falls das Video nicht abgespielt werden kann.
- ② Mit dem `source`-Element innerhalb des `video`-Elements können Sie den Namen einer alternativ abzuspielenden Videodatei angeben, falls der Browser das Dateiformat der direkt im `video`-Element angegebenen Datei nicht kennt. Sie könnten hier auch mehrere `source`-Elemente untereinander angeben.



Video im Microsoft Internet Explorer 10

Die direkt im `video`-Element angegebene Videodatei sollte ein Format haben, das der Browser auf jeden Fall erkennt. Zum Zeitpunkt der Erstellung des Buchs war unklar, welches Format sich durchsetzen wird. Sie können alternativ im Code z. B. zwei Formate gleichwertig nebeneinander angeben:

Beispiel: Video in Webseite einbetten (*kap04\video2.html*)

```
<body>
<h1>Diashow</h1>
<div>
① <video width="640" height="480" controls preload poster="diashow.jpg">
②   <source src="diashow.ogv" type="video/ogg"></source>
    <source src="diashow.mp4" type="video/mp4"></source>
</video>
</div>
</body>
```



Das Element `source` kennt zwei Attribute:

<code>type</code>	Gibt ggf. den MIME-Typ der Videodatei an, etwa <code>video/mpeg</code> , <code>video/quicktime</code> usw. Sie können das <code>type</code> -Attribut auch mit der Angabe des Codecs versehen, wenn Sie die genauen Codecs Ihrer Videodateien kennen, beispielsweise: <code>type="video/ogg; codecs=theora, vorbis"</code>
<code>media</code>	Gibt an, für welche Medientypen die Videodatei geeignet ist. Die Voreinstellung lautet: <code>Media="all"</code>

4.5 Übungen

Übung 1: Sound einbinden

Übungsdateien: *Undertone.mp3*, *Undertone.ogg* **Ergebnisdatei:** *uebung1.html*

- Binden Sie den Sound *Undertone* so in eine Webseite ein, dass er in allen modernen Browsern abgespielt werden kann.
- Fügen Sie eine Soundsteuerung ein, damit der Nutzer den Sound bei Bedarf abschalten kann.

Übung 2: Video einbinden

Übungsdateien: *Diashow.mp4*, *Diashow.ogv* **Ergebnisdatei:** *uebung2.html*

- Binden Sie das Video *Slideshow* so in eine Webseite ein, dass es in allen modernen Browsern abgespielt werden kann.
- Fügen Sie eine Grafik hinzu, die angezeigt wird, falls die Videodatei nicht abgespielt werden kann.
- Das Video soll bereits während der Übertragung abgespielt werden.

5 JavaScript

In diesem Kapitel erfahren Sie

- ✓ die grundlegenden Schritte, um JavaScript im Browser auszuführen
- ✓ wie JavaScript-Anweisungen eingeleitet werden
- ✓ welche Datentypen und Operatoren zulässig sind
- ✓ mit welchen Konstrukten Sie ein Programm steuern

Voraussetzungen

- ✓ Grundlegende Erfahrung in einer anderen Programmiersprache
- ✓ Browser mit aktivierten JavaScript-Fähigkeiten

5.1 Grundlegendes zu JavaScript

Was ist JavaScript?

JavaScript ist eine Skriptsprache, deren Befehle in die HTML-Codezeilen Ihrer Webseiten eingefügt werden. Mit JavaScript legen Sie Aktionen fest, die beispielsweise beim Anklicken eines Hyperlinks oder bei einem anderen Ereignis, wie etwa dem Laden der Seite, ausgeführt werden.

Die Interaktivität ist ein typisches Merkmal von Webseiten. Die Interaktivität moderner Webseiten geht dabei weit über die bloße Hyperlinkfunktionalität hinaus.

Da reines HTML ausschließlich statisch ist, arbeiten manche Autoren zu diesem Zweck mit serverseitigen Skripten, die zum Beispiel in den Programmiersprachen PHP oder Perl verfasst sind. Ein solches Skript wird nicht im Browser des Betrachters, sondern immer auf einem Webserver ausgewertet. So ist es beispielsweise möglich, dem Betrachter aufgrund seiner vorherigen Besuche individualisierte Seiten zu präsentieren.

JavaScript hingegen wird stets komplett lokal im Browser des Benutzers ausgewertet. So können Sie direkt im Browser auf die Aktivitäten des Nutzers reagieren. Es sind beispielsweise die folgenden Zusatzfunktionen für Webseiten möglich, die sich mit reinem HTML nicht realisieren lassen:

- ✓ Grafiken können ausgetauscht werden, sobald der Benutzer mit der Maus darauf zeigt.
- ✓ Eingaben, die in ein Feld eines Formulars geschrieben werden, können andere Formularfelder beeinflussen oder Aktionen wie mathematische Berechnungen durchführen.
- ✓ HTML-Formulare können auf Richtigkeit aller Eingaben geprüft werden.
- ✓ Die Browserversion des Benutzers kann abgefragt werden. Damit können Webseiten so programmiert werden, dass jeder Nutzer die für sein System passende Version angezeigt bekommt.

Aber auch deutlich komplexere Webanwendungen – Spiele, Textverarbeitungsprogramme, Datenbankanwendungen und vieles mehr – lassen sich mit HTML5 und JavaScript entwickeln.

Um JavaScript zu programmieren, brauchen Sie nur einen Texteditor und einen Browser mit aktiviertem JavaScript. Es ist also keine zusätzliche Software und auch kein Webserver notwendig, auf dem das Programm gestartet wird, sondern Sie können alle Arbeiten sofort auf Ihrem Computer ausführen.



JavaScript hat jedoch auch verschiedene Nachteile, derer Sie sich bewusst sein sollten.

Deaktiviertes JavaScript	Aufgrund aufdringlicher Werbefenster (Popups) auf Webseiten oder der Möglichkeit, mit JavaScript Schadprogramme wie etwa Skriptviren zu programmieren, deaktivieren aktuelle Internetstatistiken zufolge immerhin 5-10% aller Nutzer die JavaScript-Fähigkeit ihres Browsers. Deshalb sollten Sie für die Hauptnavigation Ihres Webauftritts besser auf JavaScript verzichten.
Probleme mit der Zugänglichkeit	Viele alternative Ausgabegeräte und Suchmaschinen können JavaScript nicht richtig interpretieren.
Verschiedene Browserversionen	Die unterschiedlichen Browser gehen mit JavaScript auch unterschiedlich um, sodass es schwer ist, browserübergreifend zu programmieren. Dieses Problem lässt sich gut mit JavaScript-Frameworks umgehen.

5.2 JavaScript in Webseiten einfügen

JavaScript mit dem `script`-Element in eine Webseite einfügen

Um JavaScript in eine HTML-Webseite einzufügen, verwenden Sie das `script`-Element. Setzen Sie den JavaScript-Code zwischen das Tagpaar `<script> ... </script>`. Dieses Element teilt dem Browser mit, dass innerhalb des HTML-Dokuments Java-Script verwendet wird.

<code><script> <!-- //--> </script></code>	Diese Angabe leitet einen Abschnitt mit JavaScript-Anweisungen ein. Die Tags <code><!--</code> und <code>--></code> markieren einen Kommentar. Sie stellen damit sicher, dass Browser, die kein JavaScript interpretieren, den JavaScript-Code nicht als Text am Bildschirm darstellen. Das früher gebräuchliche Attribut <code>type="text/javascript"</code> innerhalb des <code>script</code> -Elements können Sie weglassen, da JavaScript die Standard-Skriptsprache in HTML5 und in allen modernen Browsern ist.
<code><noscript> </noscript></code>	Ist die Verwendung von JavaScript im Browser abgeschaltet oder versteht der Browser JavaScript generell nicht, werden stattdessen die Informationen angezeigt, die zwischen den Tags <code><noscript>...</noscript></code> stehen. Hier bietet sich eine alternative Seitengestaltung an, die ohne JavaScript-Funktionen auskommt.

Sie können das `script`-Element überall auf der Seite verwenden. Aus Gründen der Übersichtlichkeit und der besseren Auffindbarkeit hat es sich durchgesetzt, den JavaScript-Bereich im Dokumentenkopf zu definieren. Somit ist sichergestellt, dass der Code dem Browser bereits bekannt ist, bevor er ausgeführt werden soll.



Ausnahme: Manchmal muss ein Skriptblock an einer speziellen Stelle im Dokumentkörper stehen, damit er funktioniert.



Dabei gilt, dass ein JavaScript-Skript sofort ausgeführt wird, wenn es nicht in einer Funktion (vgl. Kapitel 6) definiert wurde. Ist der Skript-Bereich im Dokumentenkörper festgelegt, kann das Ergebnis an dieser Stelle ausgegeben werden. Beispielsweise ist dies für die Ausgabe des aktuellen Datums und der Uhrzeit im Dokument möglich.

Sie können beliebig viele Skripte in Ihrem HTML-Dokument unterbringen.



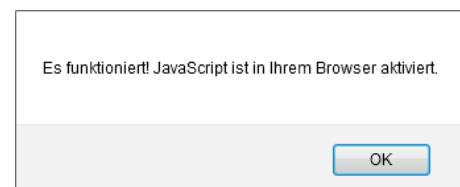
Beispiel: JavaScript-Aktivierung prüfen (*kap05\js-test.html*)

Prüfen Sie die korrekte Funktionsweise von JavaScript in Ihrem Browser, indem Sie ein HTML-Dokument erstellen, das einen JavaScript-Befehl ausführt.

Geben Sie den folgenden Quellcode ein, und speichern Sie ihn in der Datei *test.html*. Nach dem Öffnen des HTML-Dokuments in Ihrem Browser sollte sich ein Popup-Fenster öffnen.

```
<html>
<head>
<meta charset="utf-8">
<title>Der erste JavaScript-Test</title>
① <script>
②   alert('Es funktioniert! JavaScript ist in Ihrem Browser aktiviert.');
③ </script>
<noscript>
  Diese Seite verwendet JavaScript...
</noscript>
</head>
<body>
</body>
</html>
```

- ① Dem Browser wird über die Anweisung `<script>` im head des Dokuments mitgeteilt, dass nun ein Skript beginnt.
- ② Die JavaScript-Funktion `alert` zeigt ein Meldungsfenster an. Als Text wird die angegebene Zeichenkette eingeblendet.
- ③ Mit dem schließenden Tag `</script>` wird der JavaScript-Teil beendet.



JavaScript-Meldung in Firefox



Externe JavaScript-Dateien verwenden

Statt den JavaScript-Code direkt in die Webseite einzufügen, können Sie den JavaScript-Code in eine externe Textdatei mit der Dateiendung `.js` einfügen und diese Textdatei in die HTML-Seite einbinden. Dies bietet sich vor allem dann an, wenn Sie das JavaScript in mehreren Webseiten verwenden möchten. Sie verwenden zum Einbinden das Attribut `src` des Elements `script`.

Beispiel: Externes JavaScript einbinden (*kap05\extern.html*)

Geben Sie den folgenden Quellcode ein und speichern Sie ihn in der Datei *extern.html*. Nach dem Öffnen des HTML-Dokuments in Ihrem Browser sollte sich ein Popup-Fenster öffnen.

```
<html>
<body>
<script src="meinSkript.js"></script>
</body>
</html>
```

Hier wurde der Verweis auf das JavaScript-Skript `meinSkript.js` im body-Abschnitt des Dokuments platziert. Sie können es sowohl im head- als auch im body-Abschnitt einfügen.



Fertige JavaScript-Skripte aus dem Internet verwenden

Im Internet werden zahlreiche fertige JavaScript-Skripte angeboten. Sie können diese in der Regel kostenlos herunterladen und auf eine der genannten Arten in Ihre Webseite einbinden. Dies ist jedoch – gerade für Einsteiger – nur dann empfehlenswert, wenn sichergestellt ist, dass das Skript von einer seriösen Quelle stammt und welche Funktionalität es beinhaltet. Außerdem ist zu beachten, dass der Copyright-Vermerk des Autors im Code erhalten bleiben muss. Informieren Sie sich über weitere urheberrechtliche Richtlinien des Code-Anbieters, bevor Sie den Code einbinden.

5.3 Variablen und Datentypen

Die Sprache JavaScript besteht aus Variablen, Operatoren, Objekten, Eigenschaften, Funktionen usw. Die einzelnen Bestandteile werden Sie in den folgenden Abschnitten kennenlernen.

Variablen

Variablen sind nichts anderes als Behälter, die Informationen, „Werte“ genannt, zur späteren Verwendung aufzubewahren. Variablen können während der Programmausführung unterschiedliche veränderbare Werte annehmen, wie z. B. Zwischen- oder Endergebnisse aus Berechnungen. Für jede Variable wird ein Speicherplatz im Arbeitsspeicher Ihres Computers reserviert. Im Programm greifen Sie auf diesen Bereich über den Variablennamen zu.

Die Arbeit mit Variablen gehört zu den grundlegenden JavaScript-Techniken.

Eine Variable wird an einer beliebigen Stelle in einem JavaScript-Programm mithilfe der Anweisung `var` definiert, gefolgt vom Namen der Variablen. Um die Variable zu initialisieren, also mit einem Wert zu versehen, geben Sie den Zuweisungsoperator `=` und den gewünschten Wert an.

Mit dem Zuweisungsoperator `=` weisen Sie einer Variablen einen Wert zu. Die Variable steht dabei links vom Zuweisungsoperator `=`, der Wert rechts davon.



Im Folgenden werden die unterschiedlichen Möglichkeiten der Definition einer Variablen gezeigt.

Sie können eine Variable zunächst definieren, ohne ihr einen Wert zuzuweisen. Im späteren Verlauf des Programms können Sie ihr dann einen Wert zuweisen.

```
// Definition der Variablen x ohne Wertzuweisung
var x;
// spätere Wertzuweisung;
x=12;
```

Alternativ können Sie einer Variablen schon bei der Definition einen Wert zuweisen.

```
// Definition und gleichzeitige Wertzuweisung einer Zeichenkette
var _meine = 'Test';
// Definition und gleichzeitige Wertzuweisung eines Booleschen Werts
var smh12 = true;
```

Auch mehrere Variable können sie gleichzeitig definieren und ihnen Werte zuweisen. Hierbei trennen Sie die einzelnen Variablen durch Kommata voneinander.

```
var a = 2, b = 3, c = 4;
```



Ein Programm wird sicherer, wenn Variablen schon vor ihrer Verwendung definiert und initialisiert werden. So werden die Werte von außen übermittelte Variablen eventuell von der im Programm vorgenommenen Initialisierung überschrieben und es wird eine Datenmanipulation durch Dritte vermieden.



Da die Namen von Variablen, Funktionen, Objekten usw. case-sensitive sind (d. h., es wird zwischen groß- und kleingeschriebenen Buchstaben unterschieden), verweisen z. B. die Variablen `Auto` und `auto` auf unterschiedliche Speicherbereiche.

Datentypen

Daten werden ihrer Form nach in Datentypen eingeteilt. Grundsätzlich kann eine Variable innerhalb eines JavaScripts von unterschiedlichem Datentyp sein. Dies bedeutet, dass eine Variable z. B. zuerst eine Zahl sein kann, im nächsten Schritt aber zu einer Zeichenkette erweitert werden kann. Da diese Dynamik jedoch logische Fehler und unerwünschte Nebeneffekte hervorrufen kann, sollte eine Variable immer vom selben Datentyp sein.

Variablen erhalten ihren Datentyp durch die Zuweisung eines Werts. Man gibt den Datentyp also implizit an, z. B. durch die Zuweisung einer Ganzzahl.



JavaScript ist keine streng typisierte Sprache. Der Datentyp einer Variablen muss nicht bei der Deklaration angegeben werden, sondern wird bei der Wertzuweisung automatisch ermittelt und kann durch verschiedene Wertzuweisungen auch wechseln. Um Fehler zu vermeiden, sollte eine Variable immer vom selben Datentyp sein.

In JavaScript gibt es drei grundlegende Datentypen:

- ✓ Zahlen
- ✓ Zeichenketten (Strings)
- ✓ boolescher Wert (Wahrheitswerte)

und außerdem:

- ✓ Undefined (Wurde eine Variable deklariert, ihr aber kein Wert zugewiesen, erhält sie den Wert `Undefined`).
- ✓ Objekte (komplexer Datentyp, der in Kapitel 7 näher erläutert wird)

Ganze Zahlen

Ganze Zahlen besitzen keine Nachkommastellen und können negative sowie positive Werte und den Wert null annehmen. In JavaScript können sie in einer von drei möglichen Darstellungen eingesetzt werden:

Darstellung	Erläuterung	Beispiel
Dezimaldarstellung	Dezimaldarstellung: Dies ist die gebräuchlichste Darstellung von Zahlen. Die Zahl wird im Dezimalsystem (Basis 10) dargestellt, d. h., dass die Ziffern 0..9 verwendet werden.	0 15 123
Hexadezimal-darstellung	Hexadezimalzahlen sind Zahlen mit der Basis 16 und verwenden die Ziffern 0 bis 9 sowie die Zeichen A, B, C, D, E, F. A entspricht dabei dem Dezimalwert 10, F dem Dezimalwert 15. Werte dieses Datentyps werden in JavaScript durch die vorangestellten Zeichen Ox (Null und x) gekennzeichnet.	0x0 0xF 0x7B
Oktaldarstellung	Oktalzahlen sind Zahlen mit der Basis 8 und verwenden nur die Ziffern 0 bis 7. Werte dieses Datentyps werden in JavaScript durch eine vorangestellte 0 (Null) gekennzeichnet. Die Oktaldarstellung ist seit JavaScript 1.5 nicht mehr Bestandteil von JavaScript und wird nur noch aus Gründen der Rückwärtskompatibilität unterstützt.	00 017 0173

Gleitkommazahlen

Gleitkommazahlen können Nachkommastellen besitzen. Das Kennzeichen ist entweder ein Dezimalpunkt, ein Exponent oder beides. Bei einer Fixkommazahl ist die Position des Kommas vorgegeben, bei einer Gleitkommazahl kann das Komma „gleiten“.

In JavaScript wird ein Punkt `.` verwendet, um die Dezimalstellen einer Zahl abzutrennen.

Fixkommazahl	Gleitkommazahl
1000.0	1.0E3
2000.4	20004.0E-1
-4000.0	-4e3
-0.004	-4e-3

Zeichenketten

Eine Zeichenkette (String) ist eine Zeichenfolge aus Buchstaben, Zahlen und Sonderzeichen, die durch einfache oder doppelte Anführungszeichen gekennzeichnet ist. Es ist jedoch darauf zu achten, dass zur Begrenzung ein und derselben Zeichenkette dieselbe Anführungszeichenart verwendet wird: Wenn Sie eine Zeichenkette mit einem doppelten Anführungszeichen `"` beginnen, müssen Sie diese auch mit einem doppelten Anführungszeichen beenden.

Möchten Sie den Text `Und ich fragte ihn: "Hallo, wie geht es Dir?"` auf dem Bildschirm ausgeben, müssen Sie die Zeichenkette mit einem einfachen Anführungszeichen `'` beginnen und beenden, da sich die doppelten Anführungszeichen bereits in der auszugebenden Zeichenkette befinden.

Angabe in JavaScript	Ausgabe
<code>"Dies ist eine Zeichenkette."</code>	Dies ist eine Zeichenkette.
<code>'Das ist auch eine Zeichenkette.'</code>	Das ist auch eine Zeichenkette.
<code>'Die Zahl "123" ist auch eine Zeichenkette.'</code>	Die Zahl "123" ist auch eine Zeichenkette.
<code>"'So funktioniert es auch', sagte er."</code>	'So funktioniert es auch', sagte er.

Innerhalb einer Zeichenkette können Sie Steuerzeichen angeben, um beispielsweise einen Zeilenumbruch durchzuführen oder einen Tabulator einzufügen. Diese Steuerzeichen werden mit einem Backslash `\` eingeleitet und als Escape-Sequenzen bezeichnet.

Die Wirkung der Steuerzeichen ist nur innerhalb eines ausgegebenen Textes zu erkennen, z. B. innerhalb einer Meldung über die Funktion `alert` (vgl. Abschnitt 6.1). Bei der Ausgabe von Text im Browser über die Funktion `document.write()` (vgl. Abschnitt 6.2) müssen Sie stattdessen HTML-Tags verwenden, da der Browser Tabulatoren und Zeilenumbrüche im HTML-Code als Leerzeichen interpretiert.



Steuerzeichen	Bedeutung
<code>\n</code>	new line: Zeilenvorschub (neue Zeile)
<code>\r</code>	return: "Wagenrücklauf": Der Cursor steht wieder an Position 1. In JavaScript hat dies keine weitere Bedeutung.
<code>\t</code>	Tabulator
<code>\f</code>	form feed: Seitenvorschub
<code>\b</code>	backspace: ein Zeichen zurück
<code>\"</code>	doppeltes Anführungszeichen, auch innerhalb von doppelten Anführungszeichen, z. B. <code>alert ("\"") ;</code>
<code>\'</code>	einfaches Anführungszeichen, auch innerhalb von einfachen Anführungszeichen, z. B. <code>alert ('\'') ;</code>
<code>\\"</code>	Backslash

Boolesche Werte (Wahrheitswerte)

Die booleschen Werte sind `true` (wahr) und `false` (falsch). Wahrheitswerte werden eingesetzt, wenn ein Wert nur zwei Zustände annehmen kann, z. B. Licht an oder Licht aus. Ausdrücke geben häufig einen booleschen Wert zurück, z. B. beim Vergleichen von Zahlen. Der Vergleich $2 > 3$ liefert das Ergebnis `false`, weil die Zahl 2 nicht größer ist als 3.

Boolescher Wert	Bedeutung
<code>true</code>	"wahr", die Bedingung ist erfüllt, z. B. $2 < 3$
<code>false</code>	"falsch", die Bedingung ist nicht erfüllt, z. B. $2 > 3$

5.4 Operatoren

Operatoren sind Zeichen, die verwendet werden, um Berechnungen durchzuführen oder Verknüpfungen und Vergleiche zwischen Variablen herzustellen.

Eine Operation arbeitet mit einem oder mehreren Objekten (sogenannten **Operanden**), auf die der jeweilige Operator angewandt wird. Eine Operation unter Zuhilfenahme eines Operators erzeugt immer einen Ergebniswert. Manche Operatoren können nur in Verbindung mit Variablen eines bestimmten Datentyps eingesetzt werden. Es folgt eine kurze Auflistung der möglichen Operatoren in JavaScript.

Arithmetische Operatoren

Name	Operator	Syntax	Beispiel	Wert
Addition	<code>+</code>	Summand1 + Summand2	<code>7 + 4</code>	11
Subtraktion	<code>-</code>	Minuend - Subtrahend	<code>7 - 4</code>	3
Multiplikation	<code>*</code>	Faktor1 * Faktor2	<code>7 * 4</code>	28
Division	<code>/</code>	Dividend / Divisor	<code>7 / 4</code>	1.75
Modulo (Rest einer Ganz Zahldivision)	<code>%</code>	Dividend % Divisor	<code>7 % 4</code>	3
Negation	<code>-</code>	-	<code>-(2+5)</code>	-7
Inkrement (siehe Erläuterung unten)	<code>++</code>	<code>++Variable;</code> <code>Variable++</code>	<code>x=10; ++x;</code> <code>y=135; y++</code>	11 136
Dekrement	<code>--</code>	<code>--Variable;</code> <code>Variable--</code>	<code>x=10; --x;</code> <code>y=135; y--</code>	9 134



- ✓ Eine Operation unter Zuhilfenahme des **Modulo**-Operators benennt den Rest aus einer Division zweier Ganzzahlen. Da JavaScript keinen Operator für die ganzzahlige Division besitzt, müssen Sie diesen mithilfe des Modulo-Operators selbst nachbilden.
- ✓ Der **Inkrementoperator** sowie der **Dekrementoperator** sind unäre Operatoren, d. h., sie werden nur auf einen Operanden angewendet. Das Inkrement `++` bewirkt, dass der Wert des Operanden um 1 erhöht wird. Das Dekrement `--` bewirkt, dass der Wert um 1 reduziert wird. Die Angabe vor dem Operanden wird als Präfix-, die nach dem Operanden als Postfix-Notation bezeichnet. Im ersten Fall wird die Operation vor jeder weiteren Berechnung ausgeführt, in der Postfix-Notation erst nach der Berechnung des Ausdrucks.

Vergleichsoperatoren

Vergleichsoperatoren werden benutzt, um die Werte von zwei Operanden miteinander zu vergleichen. Werden Vergleichsoperatoren auf Zeichenketten angewendet, ist die Reihenfolge der Zeichen im Zeichensatz von Bedeutung. Ziffern werden z. B. niedriger eingestuft als Buchstaben und Großbuchstaben niedriger als Kleinbuchstaben.

Name	Operator	Syntax	Beispiel	Wert
gleich	<code>==</code>	<code>Operand1 == Operand2</code>	<code>2 + 1 == 4</code>	<code>false</code>
ungleich	<code>!=</code>	<code>Operand1 != Operand2</code>	<code>"Wort" != "WORT"</code>	<code>true</code>
strikte (Un-) Gleichheit (siehe Erläuterung unten)	<code>===</code> <code>!==</code>	<code>Operand1 === Operand2</code> <code>Operand1 !== Operand2</code>	<code>2 + 4 === 6</code> <code>"Hund" !== "HUND"</code>	<code>true</code> <code>true</code>
kleiner als	<code><</code>	<code>Operand1 < Operand2</code>	<code>4 < 5</code>	<code>true</code>
größer als	<code>></code>	<code>Operand1 > Operand2</code>	<code>"abd" > "abc"</code>	<code>true</code>
kleiner/gleich	<code><=</code>	<code>Operand1 <= Operand2</code>	<code>6 <= 7</code>	<code>true</code>
größer/gleich	<code>>=</code>	<code>Operand1 >= Operand2</code>	<code>"6" >= "7"</code>	<code>false</code>

Wenn Sie zwei Operanden auf strikte Gleichheit prüfen, müssen diese den gleichen Wert und den gleichen Datentyp besitzen. Wird auf einfache Gleichheit geprüft, können auch Typumwandlungen zu einer Auswertung von `true` führen. Im Falle von strikter Gleichheit ist dies nicht möglich.



```
document.write(2 == "2"); // liefert nach der Typumwandlung von "2" nach 2 true
document.write(2 === "2"); // liefert false, da unterschiedliche Typen vorliegen
```

Logische Operatoren

Im Gegensatz zu den Vergleichsoperatoren werden mit den logischen Operatoren die booleschen Wahrheitswerte `true` und `false` miteinander verknüpft. Der Ergebniswert eines logischen Ausdrucks besteht aus einem booleschen Wert.

Name	Operator	Syntax	Beispiel	Wert
UND	<code>&&</code>	<code>Operand1 && Operand2</code>	<code>true && false</code>	<code>false</code>
ODER	<code> </code>	<code>Operand1 Operand2</code>	<code>true false</code>	<code>true</code>
NICHT	<code>!</code>	<code>!Operand</code>	<code>!true</code>	<code>false</code>

Verknüpfungsoperator

Dieser Operator, der auch Konkatenationsoperator (engl.: concatenate = verknüpfen) genannt wird, verknüpft zwei Zeichenketten und liefert die zusammengesetzte Zeichenkette als Ergebniswert.

Name	Operator	Syntax	Beispiel	Wert
Verbinden	<code>+</code>	<code>Operand1 + Operand2</code>	<code>"Zimmer" + "pflanze"</code> <code>x=10; "x-Wert: " + x</code>	<code>Zimmerpflanze</code> <code>x-Wert: 10</code>



Das Zeichen für den Verknüpfungsoperator kann nicht vom Berechnungsoperator für eine Addition unterschieden werden. Deshalb sollten Sie beim Programmieren darauf achten, ob Ihre Variablen Zahlen oder Zeichenketten enthalten. Wird der Operator `+` verwendet, wird er als Konkatenationsoperator interpretiert, wenn die Variablen nicht explizit als Zahlen angegeben werden. Diese Tatsache führt zu dem Ergebnis, dass die Anweisungen `"1" + "1"` und `1 + "1"` den Wert `"11"` ergeben und nicht die Zahl 2.

Bedingungsoperator

Mithilfe des Bedingungsoperators können einige bedingungsabhängige Anweisungen verkürzt dargestellt werden.

Ausdruck ? Truewert : Falsewert



Der konditionale Operator ist eine Kurzform der nachfolgenden `if-else`-Auswahl und wird als ternärer Operator bezeichnet, weil er die folgenden drei Operanden benötigt:

- ✓ einen logischen Ausdruck, der den Wert `true` oder `false` liefert;
- ✓ einen Rückgabewert eines beliebigen Datentyps, der zurückgegeben wird, wenn der Ausdruck den Wert `true` liefert (`if`-Zweig);
- ✓ einen Rückgabewert eines beliebigen Datentyps, der zurückgegeben wird, wenn der Ausdruck den Wert `false` liefert (`else`-Zweig).

Name	Operator	Syntax	Beispiel	Wert
konditional (entweder/oder)	<code>? :</code>	<code>op1 ? op2 : op3</code>	<code>stunde > 12 ? 'P.M.' : 'A.M.'</code>	Falls <code>stunde</code> größer 12 ist, wird die Zeichenkette ' <code>P.M.</code> ', ansonsten ' <code>A.M.</code> ' geliefert.

typeof-Operator

JavaScript ist keine streng typisierte Sprache. Der Interpreter wandelt Operanden automatisch in die entsprechenden Datentypen um, damit eine Operation richtig ausgeführt wird.

Beispiel	Ergebniswert	Erklärung
<code>"Text plus Zahl" + 7</code>	<code>"Text plus Zahl 7"</code>	Konvertierung der Zahl 7 in einen String
<code>"Text plus Zahl" + 7 * 7</code>	<code>"Text plus Zahl 49"</code>	"Punkt vor Strich", dann Konvertierung
<code>7 + 7 + " Text plus Zahl"</code>	<code>"14 Text plus Zahl"</code>	Abarbeitung der Reihe nach
<code>"Text plus Zahl" + 7 + 7</code>	<code>"Text plus Zahl 77"</code>	Abarbeitung der Reihe nach
<code>"7" * 7</code>	<code>49</code>	Konvertierung der Ziffer 7 in die Zahl 7

Aufgrund der automatischen Typenkonvertierung kann es passieren, dass der aktuelle Typ einer Variablen nicht eindeutig klar ist. Der Operator `typeof` ermöglicht es, den Typ einer Variablen auszulesen. Er liefert einen der folgenden Werte:

- ✓ `number` für Zahlen,
- ✓ `string` für Zeichenketten,
- ✓ `boolean` für Wahrheitswerte,
- ✓ `undefined` für nicht initialisierte Variable.

Name	Operator	Syntax	Beispiel	Wert
typeof (VariablenTyp)	typeof	typeof Operand	typeof 12	"number"
			typeof "Zimmer"	"string"

5.5 Kontrollstrukturen

In JavaScript werden alle Anweisungen normalerweise der Reihe nach ausgeführt. Es ist jedoch wünschenswert, auf eine Eingabe des Benutzers zu reagieren. So könnte beispielsweise ein bestimmter Teil des Skripts nur dann ausgeführt werden, wenn der Benutzer eine vorangestellte Frage mit JA beantwortet hat. Ebenso wichtig kann es sein, auf die Ausgabebedingungen des Benutzers zu reagieren, beispielsweise auf die Größe des von ihm verwendeten Bildschirms und den verwendeten Browser. So ist es möglich, dem Benutzer per JavaScript eine „maßgeschneiderte“ Website-Version zu liefern

JavaScript stellt für eine bedingte Kontrollstruktur die Anweisungen `if-else` und `switch` zur Verfügung und für eine wiederholte Ausführung von Befehlen `for`-Schleife und die `while`-Schleife.

Die Beispiele zu den nachfolgenden Kontrollstrukturen finden Sie in der Datei `kap05\steuerung.html`.

Bedingte Auswahl

Diese Kontrollstruktur erlaubt es Ihnen, bestimmte JavaScript-Anweisungen nur unter gewissen Umständen ausführen zu lassen. Dies lässt sich mit der bedingten Anweisung `if-else` (*wenn dann... ansonsten...*) realisieren.

<code>if (Bedingung) { // Anweisungen }</code>	Trifft die in Klammern angegebene Bedingung zu, werden die innerhalb der geschweiften Klammern stehende Skriptblock mit einer oder mehreren Anweisungen ausgeführt. Wird die Bedingung nicht erfüllt, werden die Anweisungen übersprungen.
<code>if (Bedingung) { // Anweisungen A } else { // Anweisungen B }</code>	Dies ist die erweiterte Form der bedingten Anweisung. Im Gegensatz zur vorherigen Abfrage werden bei Nichterfüllung der Bedingung die Anweisungen ausgeführt, die nach der <code>else</code> -Anweisung angegeben sind.

Beispiele

Eine Zahl wird über den Modulo-Operator `%` darauf überprüft, ob bei der Ganzzahl-Division von 2 ein Rest übrig bleibt. Wenn nicht, ist die Zahl gerade und eine entsprechende Meldung wird ausgegeben.

```
if (x % 2 == 0) {  
    alert(x + ' ist eine gerade Zahl.');
```

Wenn die Zahl ungerade ist, wird ebenfalls darauf hingewiesen.

```
if (x % 2 == 0) {
    alert(x + ' ist eine gerade Zahl.');
} else {
    alert(x + ' ist eine ungerade Zahl.');
}
```

Fallauswahl

Bei einer Fallauswahl, die auch Selektion heißt, wird der Wert einer Variablen ausgewertet und in Abhängigkeit von diesem Wert eine Anweisung bzw. ein Anweisungsblock ausgeführt. Die Variable, deren Inhalt geprüft werden soll, wird auch als Selektor bezeichnet.

Möchten Sie zwischen mehreren Fällen unterscheiden und die Ergebnisse entsprechend auswerten, verwenden Sie statt **if-else** die Schlüsselwörter **switch** und **case**.

<pre>switch (Variable) { case Wert1: Anweisungen; break; case Wert2: Anweisungen; break; ... default: Anweisungen; break; }</pre>	<ul style="list-style-type: none"> ✓ switch und der in Klammern stehende und zu überprüfende Selektor leiten einen Anweisungsblock ein, der mehrere case-Anweisungen enthalten kann. ✓ Innerhalb der switch-Abfrage werden die verschiedenen case-Abfragen definiert. Diese beinhalten den möglichen Wert der Bedingung. Stimmt ein Wert des Selektors mit dem Wert einer case-Abfrage überein, werden die entsprechenden Anweisungen nacheinander ausgeführt. ✓ Über das Schlüsselwort break beenden Sie die Ausführung des aktuellen Anweisungsblocks und somit auch die switch-Abfrage. ✓ Mit default definieren Sie Anweisungen, die ausgeführt werden, wenn keine der vorherigen case-Abfragen mit dem Selektor übereinstimmt. Die Angabe von default ist optional.
--	--

Beispiel

```
switch (Angabe)
{
    case "Gewicht": Einheit = "kg"; break;
    case "Länge" : Einheit = "km"; break;
    case "Zeit"   : Einheit = "s";  break;
    default       : Einheit = "";   break;
}
```

In Abhängigkeit vom Wert der Variable **Angabe** werden die einzelnen Wertzuweisungen für die Variable **Einheit** durchgeführt. Hat die Variable **Angabe** den Wert **Gewicht**, erhält die Variable **Einheit** den Wert **kg** usw. Trifft keine der drei **case**-Abfragen zu, wird der Variablen **Einheit** über den **default**-Zweig eine leere Zeichenkette zugewiesen.

Schleifen

Oftmals müssen die gleichen Anweisungen mehrmals in einer Schleife wiederholt werden. Meistens ist dabei auch nicht vorhersehbar, wie oft diese Schleifen ausgeführt werden müssen.

Die Schleifendurchläufe werden entweder gezählt, wenn die Anzahl vom Autor festgelegt wird/werden kann, oder in Abhängigkeit einer Bedingung ausgeführt.

Zählergesteuerte Schleifen

Diese Kontrollstruktur erlaubt eine genau festgelegte Anzahl von Befehlwiederholungen. Diese Art der Steuerung wird auch Zählschleife genannt.

<pre><code>for (Initialisierung; Bedingung; Aktualisierung;) { // Anweisung(en) }</code></pre>	<p>Bei der Bearbeitung einer <code>for</code>-Schleife wird an erster Position beim ersten Schleifendurchlauf die Initialisierung der Zahlvariablen durchgeführt, indem ihr ein bestimmter Wert zugewiesen wird.</p> <p>An zweiter Stelle wird überprüft, ob eine angegebene Bedingung erfüllt ist. Ist sie nicht erfüllt, wird die Schleife beendet. Andernfalls wird der Anweisungsblock ausgeführt.</p> <p>Im letzten Parameter wird die Reininitialisierung der Variablen vorgenommen, z. B. Hoch- oder Herunterzählen des Variablenwertes.</p> <p>Das Überprüfen der Schleifenbedingung, das Ausführen der Anweisungen und die anschließende Aktualisierung werden so lange wiederholt, bis die Schleifenbedingung nicht mehr erfüllt ist und die Schleife verlassen wird.</p>
--	---

Beispiel

```
for ( var i=0; i < 10; i++ ) {
    alert(i);
}
```

Die Schleife beginnt mit der Initialisierung der Variablen `i`. Sie erhält den Wert 0. Die Schleife wird so lange durchlaufen, wie der Wert `i` kleiner 10 ist. Bei der Aktualisierung wird der Wert der Variablen `i` jedes Mal um 1 erhöht (`i++`). Somit werden über den Befehl `alert` die Zahlen 0 bis 9 ausgegeben.

Kopf- und fußgesteuerte Schleifen

Kopf- und fußgesteuerte Schleifen setzen Sie ein, wenn die Ausführung von einer bestimmten Bedingung abhängig ist.

Bei einer **kopfgesteuerten Schleife** wird vor dem Eintritt in die Schleife eine festgelegte Bedingung geprüft. Diese Programmsteuerung leiten Sie mit dem Befehl `while` ein.

<pre><code>while (Bedingung) { // Anweisung(en) }</code></pre>	<p>Die kopfgesteuerte Wiederholung wird durch das Schlüsselwort <code>while</code> (solange) eingeleitet. Danach folgt eine Bedingung, die am Anfang der Schleife prüft, ob die Schleife durchlaufen werden soll.</p> <p>Die Schleife wird ausgeführt, solange die Bedingung zutrifft. Sollte die Bedingung nicht zutreffen, wird die Schleife nicht ausgeführt.</p> <p>Falls die Bedingung einen Wert <code>false</code> zurückliefert, wird die kopfgesteuerte Schleife beendet.</p>
--	--

Beispiel

```
var i=10;
while ( i > 0 ) {
    i--; alert(i);
}
```

Die Schleife wird so lange durchlaufen, wie der Wert der Variablen der Bedingung `i > 0` zutrifft. Der jeweilige Wert wird mit dem Befehl `alert` angezeigt.

Bei einer **fußgesteuerten Schleife** findet die Bedingungsprüfung am Ende der Schleife statt. Erst nach Abarbeitung der Anweisungen wird am Ende der Schleife die Bedingung daraufhin geprüft, ob die Schleife noch einmal durchlaufen werden soll oder nicht. Solange die Bedingung erfüllt ist, werden die vorausgehenden Anweisungen wiederholt ausgeführt. Dies heißt, dass die Schleife auf jeden Fall mindestens einmal durchlaufen wird.

<code>do { // Anweisung(en) } while (Bedingung)</code>	Die fußgesteuerte Schleife wird mit dem Befehl <code>do</code> eingeleitet. Danach folgt direkt die der auszuführende Anweisungsblock. Am Schleifenende erfolgt die Bedingungsprüfung über die Anweisung <code>while</code> . Die Schleife wird ausgeführt, solange die Bedingung zutrifft. Die Anweisungen innerhalb der Schleife werden mindestens einmal ausgeführt.
--	---

Beispiel

```
var i=10;  
do {  
    i--; alert(i);  
} while (i > 0)
```

Die Schleife wird mindestens einmal und so lange durchlaufen, wie der Wert der Variablen der Bedingung `i >= 0` genügt. Dieses Beispiel zeigt Ihnen, dass sich der Variablenwert ändern sollte, damit die Schleife nicht endlos ausgeführt wird.

Schleifensteuerung

Soll eine Schleife in Abhängigkeit von einer Bedingung verlassen oder aber mit der Prüfung der Bedingung fortgesetzt werden, verwenden Sie die Anweisungen `break` oder `continue`.

<code>break</code>	Das Schlüsselwort <code>break</code> kann in den Anweisungen <code>while</code> , <code>do</code> und <code>for</code> angewendet werden und beendet vorzeitig die gesamte Schleife. Es stoppt den aktuellen Schleifendurchlauf und beginnt mit der Anweisung, die der abgebrochenen Schleife unmittelbar folgt. Dazu müssen Sie in einer Schleife eine <code>if</code> -Abfrage notieren. Trifft die Bedingung zu, wird mit dem Befehl <code>break</code> die Schleife verlassen.
--------------------	--

Beispiel

```
for ( var i=0; i < 10; i++ ) {  
    if (i==5) {  
        break;  
    }  
    alert(i);  
}
```

Die `for`-Schleife richtet sich nach dem Wert der Variablen `i` und beginnt mit 0. Sie soll durchlaufen werden, bis der Wert dieser Variablen 9 beträgt. Die Schleife wird allerdings vorzeitig beendet, wenn die Variable `i` den Wert 5 erreicht hat. Die Ausgabe der Werte erfolgt damit nur von 0 bis 4.

continue	Den Befehl <code>continue</code> können Sie in den Anweisungen <code>while</code> , <code>do</code> und <code>for</code> einsetzen. Bei <code>while</code> und <code>do</code> wird die Prüfung der Bedingung durchgeführt, in der <code>for</code> -Schleife wird die Zählvariable neu ausgewertet. Im Gegensatz zu <code>break</code> wird mit <code>continue</code> nur der aktuelle Schleifendurchlauf abgebrochen.
----------	---

Beispiel

```
for ( var i=0; i < 10; i++ ) {
  if (i % 2 == 0) {
    continue;
  }
  alert(i);
}
```

Es werden nur die ungeraden Zahlen zwischen 0 und 9 ausgegeben.

Im Gegensatz zur `break`-Anweisung kann ein falsch platziertes `continue` zu Endlosschleifen führen. Deshalb sollten Sie es mit Vorsicht benutzen.



5.6 Kommentare

In JavaScript gibt es zwei Formen von Kommentaren. Die erste Form beschreibt einen einzeiligen und die zweite einen mehrzeiligen Kommentar.

//	Der doppelte Schrägstrich teilt dem Browser mit, dass alle folgenden Zeichen bis zum Ende der Zeile Kommentare sind, die nicht ausgewertet werden.
/* */	Alternativ können Sie Kommentare zwischen die Zeichen <code>/*</code> und <code>*/</code> setzen. Dabei kennzeichnet <code>/*</code> den Anfang und <code>*/</code> das Ende des Kommentars. Sie geben zunächst die Zeichen <code>/*</code> ein und beginnen den Kommentartext in der nächsten Zeile. Zuletzt geben Sie die schließenden Zeichen <code>*/</code> ebenfalls in eine eigene Zeile ein. Für mehrzeilige Kommentare verwenden Sie stets diese Schreibweise.

Beispiel

```
// Diese Zeile wird als Kommentar betrachtet.
/*
Mithilfe der beiden umschließenden Zeichen können Sie
sogar mehrzeilige Kommentare notieren, ohne dass diese
als JavaScript-Befehle betrachtet werden.
*/
```

Sparen Sie nicht mit Kommentaren. Benutzen Sie diese, um die Arbeitsweise von Skripten zu dokumentieren. So werden Sie oder Ihre Kollegen auch nach einiger Zeit noch wissen, was Sie mit den Anweisungen erreichen wollten.



5.7 Übungen

Übung 1: Werte vergleichen

Übungsdatei: --**Ergebnisdatei:** uebung1.html

1. Schreiben Sie ein Programm, das zwei Ganzzahlen vergleicht und anhand eines Vergleichsoperators zeigt, welche die Größere von beiden ist. Verwenden Sie für den Vergleich die if-else-Anweisung.

Übung 2: Summe eines Zahlenbereichs bilden

Übungsdatei: --**Ergebnisdatei:** uebung2.html

1. Berechnen Sie mithilfe der festgelegten Wiederholung die Summe der Zahlen von 1 bis 10.

Übung 3: Ermitteln, wie oft eine Zahl teilbar ist

Übungsdatei: --**Ergebnisdatei:** uebung3.html

1. Schreiben Sie ein Programm, das zu einer beliebigen Zahl im Bereich von 1 bis 1000 ermittelt, wie oft diese Zahl durch 5 teilbar ist. Verwenden Sie hierfür die kopfgesteuerte Schleife und den arithmetischen Operator Modulo (%) für die Abfrage.
2. Erklären Sie, warum Sie keine fußgesteuerte Schleife verwenden können.

6 Funktionen und Interaktionen in JavaScript

In diesem Kapitel erfahren Sie

- ✓ wie Sie verschiedene Dialogfenster einblenden
- ✓ wie Sie Funktionen erstellen und aufrufen
- ✓ wie Sie Ereignisse abfragen und darauf reagieren

Voraussetzungen

- ✓ Datentypen und Programmsteuerung

6.1 Funktionen

Um den Programmieraufwand zu verringern und Anweisungen in funktionell zusammengehörige Blöcke zu unterteilen, können Sie Funktionen verwenden. Funktionen sind eigenständige Skriptblöcke, die gemäß einer vorgegebenen Syntax zur späteren Verwendung gespeichert wurden und von Anweisungen direkt aus dem Skript aufgerufen werden.

Eine Funktion wird ausgeführt, sobald sie aufgerufen wird. Die Funktion kann direkt aufgerufen werden, wenn ein bestimmtes Ereignis eintritt – ein Benutzer beispielsweise auf eine Schaltfläche klickt – oder sie kann durch JavaScript-Code aufgerufen werden.

Anstatt also wiederholt denselben Code anzugeben, wird er in eine Funktion ausgelagert und kann über den Funktionsaufruf beliebig oft aufgerufen werden.



Bei Bedarf können Sie sich eine eigene Bibliothek mit einmal entwickelten und getesteten Funktionen anlegen. Diese können Sie dann in Ihren Dokumenten immer wieder verwenden.

```
function FktName() {  
    // Anweisungen  
}
```

Mit `function` leiten Sie eine Funktion in JavaScript ein. Danach geben Sie für die Funktion einen Namen an, mit dem Sie das Unterprogramm auf einer Webseite aufrufen können. Dabei sollten Sie einen Namen vergeben, der den Sinn der Funktion wiedergibt. So könnten Sie z. B. eine Funktion, die eine Fläche eines Kreises berechnet, mit `Kreisberechnung` benennen. Der Name einer Funktion muss mit einem Unterstrich `_` oder einem Buchstaben beginnen. Der Name kann auch Zahlen, jedoch keine Sonderzeichen enthalten. Beispielsweise: `_betrag`, `Tangente`, `Kreis2` usw. In ein direkt folgendes Klammerpaar werden die Argumente als Werte für die Funktion notiert. Die beiden geschweiften Klammern umschließen die Anweisungen, die bei Aufruf der Funktion abgearbeitet werden sollen.

Beispiel

Es wird eine JavaScript-Funktion `meineFunktion` erzeugt, die beliebige weitere Anweisungen beinhalten kann.

```
<script>  
    function meineFunktion() {  
        // die Anweisungen  
    }  
</script>
```

Funktionen aufrufen

Nach der Funktionsdefinition kann die Funktion per Funktionsaufruf ausgeführt werden. Funktionen werden mit dem Namen aufgerufen, mit dem sie definiert sind. Wenn die Funktion aufgerufen wird, werden die Anweisungen in dieser Funktion abgearbeitet und Ergebnisse über das Schlüsselwort `return` dem Aufrufer übergeben.

Funktionen werden nur dann ausgeführt, wenn sie explizit aufgerufen werden.



Ereignisse

JavaScript stellt Ihnen zahlreiche Ereignisse zur Verfügung. Tritt ein bestimmtes Ereignis ein, wird die angegebene Funktion aufgerufen und die enthaltenen Anweisungen werden abgearbeitet (**Event-Handling**).

Die am meisten verwendete Abfrage ist, ob mit der Maus auf ein bestimmtes Element geklickt wurde.

onClick=	<p>Mit <code>onClick</code> (<i>beim Klick</i>) können Sie abfragen, ob ein Besucher mit der Maus auf ein bestimmtes Element klickt.</p> <p>Dieses Ereignis kann in folgenden HTML-Elementen abgefragt werden:</p> <pre><a> <abbr> <acronym> <address> <area> <big> <blockquote> <body> <button> <caption> <center> <cite> <code> <col> <colgroup> <dd> <dfn> <dir> <div> <dl> <dt> <fieldset> <form> <h1> <h2> <h3> <h4> <h5> <h6> <hr> <i> <input> <ins> <kbd> <label> <legend> <link> <map> <menu> <noframes> <noscript> <object> <optgroup> <option> <p> <pre> <q> <s> <samp> <select> <small> <strike> <sub> <sup> <table> <tbody> <td> <textarea> <tfoot> <th> <thead> <tr> <tt> <u> <var></pre>
-----------------	--

Die Ereignisabfrage und der dazugehörige Funktionsaufruf erfolgen innerhalb eines HTML-Tags:

```

```

Beispiel: Ereignisabfrage und Funktionsaufruf (*kap06\ funktion1.html*)

Im nachfolgenden Beispiel wird die Funktion `meldung` aufgerufen, sobald der Anwender innerhalb eines Formulars auf die Schaltfläche mit der Beschriftung FUNKTION AUFRUFEN klickt.

<pre> <html> <head> <meta charset="utf-8"> <title>Funktionen in JavaScript</title> <script> function meldung() { alert("Sie haben auf die Schaltfläche geklickt."); } </script> </head> <body> <h3>Funktionsaufruf in JavaScript</h3> <form> <input type="button" value="Funktion aufrufen" onClick="meldung()"> </form> </body> </html> </pre>	<p>①</p> <p>②</p>
---	-------------------

- ① Innerhalb der Funktion `meldung` legen Sie fest, dass über den Befehl `alert` eine Meldung ausgegeben werden soll. Durch die Festlegung als Funktion wird die Meldung nicht sofort beim Laden der Webseite angezeigt.
- ② Erst beim Klick auf die Schaltfläche wird der Aufruf über das Ereignis `onClick` ausgelöst und die hinterlegte Funktion `meldung` aufgerufen und ausgeführt.

Sie haben auf die Schaltfläche geklickt.

OK

Ergebnis des JavaScript-Aufrufs nach dem Ereignis onClick



Beachten Sie die Groß-/Kleinschreibung, wenn Sie JavaScript-Funktionen verwenden. Der Funktionsaufruf muss mit derselben Groß-/Kleinschreibung erfolgen wie die Definition des Funktionsnamens.

Es gibt noch eine Vielzahl anderer Ereignisse, bei denen Sie eine Funktion aufrufen können. Die Bezeichnungen der Ereignisse können beliebig groß- oder kleingeschrieben werden, da es sich nicht um JavaScript-Code handelt. Beachten Sie, dass die Browser diese Ereignisse nicht im gleichen Umfang in allen HTML-Tags unterstützen. Eine Webseite mit den Implementierungen der verschiedenen Ereignisse finden Sie unter [kap06\onEreignis.html](#).

Ereignis	Wird ausgelöst ...	Nach HTML 5.0 erlaubt in ...
<code>onAbort</code>	bei Abbruch des Ladens einer Webseite	<code></code>
<code>onBlur</code>	beim Verlassen eines Elements	<code><a> <area> <button> <input> <label> <select> <textarea></code>
<code>onChange</code>	bei Änderungen von Angaben	<code><input> <select> <textarea></code>
<code>onDoubleClick</code>	beim doppelten Anklicken	in fast allen HTML-Tags
<code>onError</code>	im Fehlerfall (z. B. falsche Bildangabe)	<code></code>
<code>onFocus</code>	beim Aktivieren eines selektierbaren Elements	<code><a> <area> <button> <input> <label> <select> <textarea></code>
<code>onKeyDown</code>	beim Drücken einer Taste	in fast allen HTML-Tags
<code>onKeyPress</code>	beim Drücken einer Taste, die einen Zeichencode erzeugt (für ein darstellbares Zeichen steht)	in fast allen HTML-Tags
<code>onKeyUp</code>	nach dem Loslassen einer Taste	in fast allen HTML-Tags
<code>onLoad</code>	beim Laden einer Webseite	<code><frameset> <body></code>
<code>onMouseDown</code>	beim Betätigen der Maustaste	in fast allen HTML-Tags
<code>onMouseOut</code>	beim Verlassen eines Elements mit der Maus	in fast allen HTML-Tags
<code>onMouseOver</code>	beim Überfahren eines Elements mit der Maus	in fast allen HTML-Tags
<code>onMouseUp</code>	nach dem Loslassen der Maustaste	in fast allen HTML-Tags
<code>onReset</code>	beim Zurücksetzen eines Formulars	<code><form></code>
<code>onSelect</code>	beim Selektieren von Text in Eingabefeldern	<code><input> <textarea></code>
<code>onSubmit</code>	beim Absenden von Formulardaten	<code><form></code>
<code>onUnload</code>	beim Verlassen einer Webseite	<code><frameset> <body></code>

Mehrere Ereignisse gleichzeitig abfragen

Innerhalb eines Elements können Sie mehrere Ereignisse, wie `onLoad` (beim Laden) und `onUnload` (beim Verlassen der Webseite), abfragen. Dazu geben Sie die Ereignisse hintereinander an. Tritt eines von beiden möglichen Ereignissen ein, wird die entsprechende Funktion aufgerufen bzw. der entsprechende JavaScript-Code abgearbeitet.

Beispiel: Meldung beim Laden und Verlassen der Webseite (*kap06\ereignis.html*)

Als Beispiel soll beim Laden und beim Verlassen einer Webseite jeweils eine entsprechende Meldung angezeigt werden.

```

<html>
<head>
<meta charset="utf-8">
<title>Ereignisabfragen</title>
<script>
①   function meldung1() {
    alert("Ereignis: OnLoad\nHerzlich willkommen auf meiner Webseite!");
}
②   function meldung2() {
    alert("Ereignis: OnUnload\nSie wollen schon gehen? Vielen Dank für Ihren
          Besuch...") ;
}
</script>
</head>
③ <body onLoad="meldung1()" onUnLoad="meldung2()">
    <h3>Mehrere Ereignisabfragen</h3>
    <p>Im &lt;body&gt; sind die Ereignisse OnLoad und OnUnload definiert.<br>
    ④ <a href="weiterleit.html">Verlassen Sie die Seite</a> mit einem Klick auf
        den Hyperlink, um das Ereignis OnUnLoad auszuführen.</p>
</body>
</html>

```

- ① Die Funktion `meldung1()` dient dazu, beim Laden der Webseite aufgerufen zu werden.
- ② Bei Funktion `meldung2()` soll eine Meldung beim Verlassen der Webseite eingeblendet werden.
- ③ Im Tag `<body>` geben Sie die Ereignisabfragen als Attribute an. Damit stellen Sie sicher, dass die beiden Ereignisse eingelesen werden, auch wenn der Besucher den Ladevorgang der Webseite abbrechen sollte. Das Abbrechen des Ladevorgangs macht natürlich erst bei sehr ladeintensiven Webseiten Sinn.
- ④ Der Hyperlink dient dazu, die aktuelle Webseite zu verlassen und damit das Ereignis `onUnLoad` auszulösen.

Funktion mit Parameterübergabe

Im vorherigen Beispiel werden zwei verschiedene Texte am Bildschirm ausgegeben, einer zur Begrüßung und einer zur Verabschiedung. Dies wird über die beiden Funktionen `meldung1()` und `meldung2()` realisiert. Eigentlich liefern beide Funktionen dasselbe Ergebnis, indem sie eine Dialogbox anzeigen. Das Einzige, was sich ändert, ist der angezeigte Text. Günstiger ist es, Sie definieren eine Funktion, die verschiedene Werte, in diesem Fall den anzugegenden Text, verarbeiten kann. Dazu übergeben Sie der Funktion den jeweiligen Text als Parameter.

Ereignis="Funktionsname (Wert)"

Aufruf der Funktion mit Wertübergabe

```

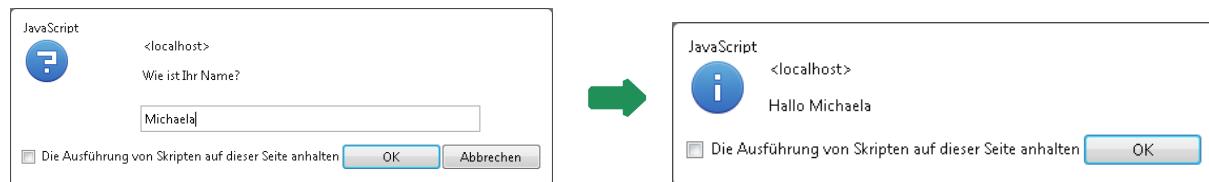
function Funktionsname (Wert)
{
    // JavaScript-Anweisungen
}
```

Aufbau der Funktion mit Wertübernahme

Die nach dem Funktionsnamen in Klammern festgelegte Variable erhält bei jedem Funktionsaufruf den Wert, der der Funktion übergeben wird. Innerhalb der Funktion können Sie nun statt des Textes immer die Variable angeben.

Beispiel

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Parameterübergabe</title>
    <script>
①      function namensausgabe(nameAusgeben) {
        alert("Hallo " + nameAusgeben);
    }
    </script>
  </head>
  <body>
    <script>
      var eingabe;
      eingabe = prompt("Wie ist Ihr Name?", "");
②      namensausgabe(eingabe)
    </script>
  </body>
</html>
```



- ① Hier wird die Funktion namensausgabe definiert. Diese erwartet den Parameter nameAusgeben.
- ② Beim Aufruf der Funktion wird kein fester Wert als Parameter in die Klammern gesetzt, sondern es wird eine Variable angegeben. In diesem Fall gibt der Benutzer seinen Namen ein, der in der Variablen eingabe gespeichert wird. Diese Variable wird als Argument an die Funktion namensausgabe übergeben (in den Parameter nameAusgeben geschrieben).



Die Übergabe von Text an eine Funktion muss immer in einfache oder doppelte Hochkommata gesetzt werden. Numerische Werte werden ohne Angabe von Hochkommata übergeben. Setzen Sie einen numerischen Wert in Hochkommata, wird er als Text behandelt.

```
meldung('Willkommen'); // Übergabe eines Textes
meldung('123'); // Übergabe eines Textes
meldung(123); // Übergabe eines Zahlenwerts
```

Beispiel: Meldung beim Laden und Verlassen der Webseite 2 (*kap06/funktion2.html*)

Das JavaScript-Beispiel, das beim Laden und Verlassen einer Webseite eine Meldung anzeigen soll, beinhaltet durch nur noch eine Funktion.

```

<html>
<head>
    <meta charset="utf-8">
    <title>Ereignisabfragen</title>
    <script>
        ①   function meldung(anzeige) {
            alert(anzeige);
        }
    </script>
</head>
② <body onLoad="meldung('Ereignis: OnLoad\nHerzlich willkommen auf meiner
    Webseite!')" onUnLoad="meldung('Ereignis: OnUnload\nSie wollen schon gehen?
    Vielen Dank für Ihren Besuch...')">
    <h3>Mehrere Ereignisabfragen</h3>
    <p>Im &lt;body&gt; sind die Ereignisse OnLoad und OnUnload definiert.<br>
    <a href="weiterleit.html">Verlassen Sie die Seite</a> mit einem Klick auf den
        Hyperlink, um das Ereignis OnUnLoad auszuführen.
</body>
</html>

```

- ① Es wird nur noch eine Funktion benötigt. Die Funktion `meldung` erhält den variablen Text und kann ihn über die entsprechende Variable `anzeige` ausgeben.
- ② Bei den Ereignissen `onLoad` und `onUnLoad` wird nur noch die Funktion `meldung` aufgerufen. Nur die übergebenen Texte unterscheiden sich voneinander.

javascript:

Das Schlüsselwort `javascript`, gefolgt von einem Doppelpunkt (:), erlaubt Ihnen das Ausführen von einzelnen JavaScript-Befehlen direkt von einem Verweis aus. Deshalb ist dieser Befehl nur in Verbindung mit dem HTML-Tag `` möglich. Sie benötigten keine `script`-Tags, sondern setzen den Befehl mitten in den HTML-Code.

javascript:	Geben Sie <code>javascript:</code> und anschließend einen JavaScript-Befehl ein, der nach einem Klick auf den Verweis ausgeführt werden soll.
--------------------	---

Beispiel

Nach dem Klick auf den Hyperlink wird ein Dialogfenster mit dem angegebenen Text angezeigt. Der Hyperlink verweist auf keine separate Seite und wird daher auch nicht ausgelöst. Die aktuelle Webseite bleibt im Browser stehen.

```
<a href="javascript:alert('Sie haben diesen Dialog von einem Verweis aus
    geöffnet.')">Dialogfenster öffnen</a>
```

6.2 Objektunabhängige Funktionen

Es gibt in JavaScript eine Reihe von vordefinierten Funktionen, die als objektunabhängige Funktionen bezeichnet werden. Diese können von Ihnen jederzeit aufgerufen werden.

Auswahl objektunabhängiger Funktionen

<code>eval()</code>	Der Funktion <code>eval</code> (<i>evaluate = auswerten</i>) kann eine Zeichenkette mit Rechenoperationen übergeben werden, deren Ergebnis zurückgeliefert wird. Enthält die Zeichenkette Zeichen, die nicht als Rechenoperation interpretiert werden können, wird eine Fehlermeldung zurückgeliefert.
<code>escape()</code> <code>unescape()</code>	<code>escape</code> liefert den hexadezimalen Wert der Steuerzeichen mit dem ASCII-Wert von 0 bis 31 sowie von Sonderzeichen wie etwa den deutschen Umlauten zurück. <code>unescape</code> führt das Gegenteil aus. Bei Angabe eines hexadezimalen Wertes wird das entsprechende Steuerzeichen zurückgegeben.
<code>isNaN()</code>	Mit der Funktion <code>isNaN</code> (<i>is Not a Number = ist keine Zahl</i>) haben Sie die Möglichkeit, zu testen, ob eine übergebener Wert keine Zeichenkette ist. Ist der Wert eine Zahl, wird <code>false</code> zurückgeliefert, ist sie keine Zahl, wird <code>true</code> zurückgeliefert.
<code>Number()</code>	Hiermit können Sie den Inhalt eines Objekts in eine Zahl umwandeln. Ist die Konvertierung nicht erfolgreich, wird der Wert <code>NaN</code> (<i>Not a Number = keine Zahl</i>) ausgegeben.
<code>parseFloat()</code>	Wandelt eine Zeichenkette in eine Zahl um und gibt diese als numerischen Wert zurück. Diese Funktion gibt <code>NaN</code> (<i>Not a Number = keine Zahl</i>) zurück, wenn die Zeichenkette mit einem nicht numerischen Zeichen beginnt. Befindet sich in der Zeichenkette ein ungültiges Zeichen, wird die Zahl bis zu diesem Zeichen zurückgegeben.
<code>parseInt()</code>	Wandelt eine Zeichenkette in eine ganze Zahl um. Diese Funktion gibt <code>NaN</code> (<i>Not a Number = keine Zahl</i>) zurück, wenn die Zeichenkette mit einem nicht numerischen Zeichen beginnt. Enthält die Zeichenkette ein ungültiges Zeichen, wird die Zahl bis zu diesem Zeichen zurückgegeben.
<code>String()</code>	Diese Funktion wandelt den Inhalt eines Objekts in eine Zeichenkette um.

Beispiel: Erklärung der objektunabhängigen Funktionen (*kap06\objunFunktion.html*)

Folgendes Beispiel erklärt die Art und Weise der verschiedenen Funktionen.

Dabei wird die Funktion `document.writeIn()` verwendet. Mit dieser wird eine Ausgabe im HTML-Dokument erzeugt. Versteht der Browser JavaScript, wird er den angegebenen Text anzeigen.

	<pre><html> <head> <meta charset="utf-8"> <title>Objektunabhängige Funktionen</title> </head> <body> <script> ① document.writeln('<p>isNaN("12345") : ' + isNaN("12345")); document.writeln('
isNaN("abcde") : ' + isNaN("abcde")); ② document.writeln('<p>Number("123asd") : ' + Number("123asd")); document.writeln('
Number("123") : ' + Number("123")); </pre>
--	---

```

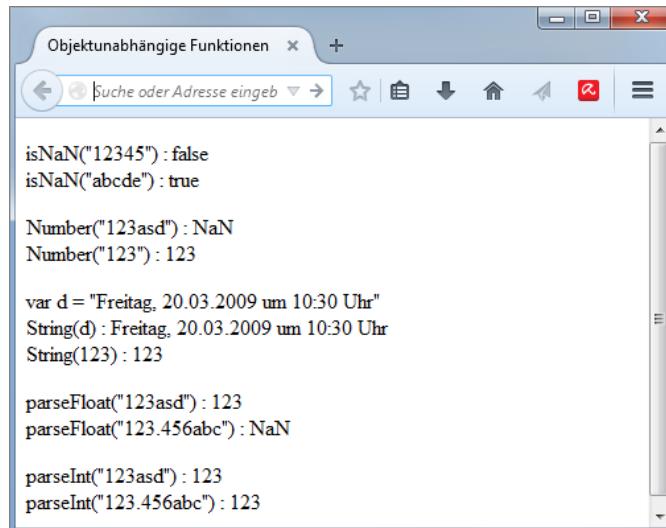
③ var d = 'Freitag, 20.03.2009 um 10:30 Uhr';
document.writeln('<p>var d = \'Freitag, 20.03.2009 um 10:30 Uhr\';');
document.writeln('<br>String(d) : ' + String(d));
document.writeln('<br>String(123) : ' + String(123));

④ document.writeln('<p>parseFloat("123asd") : ' + parseFloat("123asd"));
document.writeln('<br>parseFloat("123.4abc") : ' + parseFloat("123.4abc"));

⑤ document.writeln('<p>parseInt("123asd") : ' + parseInt("123asd"));
document.writeln('<br>parseInt("123.456abc") : ' + parseInt("123.456abc"));
</script>
</body>
</html>

```

- ① Zuerst werden die beiden Zeichenketten daraufhin untersucht, ob sie in eine Zahl umgewandelt werden können.
- ② Dann werden zwei Zeichenketten in Zahlen umgewandelt.
- ③ Die übergebenen Werte werden als Zeichenketten interpretiert.
- ④ Es wird versucht, die angegebenen Werte in eine Gleitkommazahl umzuwandeln, bis ein Fehler auftritt.
- ⑤ Mit `parseInt` wird versucht, Zeichenketten in Ganzzahlen umzuwandeln.



Rückgabewerte der objektunabhängigen Funktionen

6.3 Interaktionen

Dialogfenster

Ein Dialogfenster ist ein Fenster, das eine Meldung ausgibt. Der Benutzer kann diese Meldung mit einem Klick auf die Schaltfläche *OK* bestätigen.

<code>alert("im Dialogfenster anzuzeigen Text");</code>	Die Funktion <code>alert (Alarm)</code> öffnet das Dialogfenster. Innerhalb der Anführungszeichen können Sie einen Text oder einen Variableninhalt angeben, der angezeigt werden soll. Dabei können Sie Escape-Sequenzen verwenden: Möchten Sie beispielsweise einen Zeilenumbruch einfügen, dann geben Sie innerhalb des Textes das Sonderzeichen <code>\n</code> an.
---	--

Beispiel: Meldungsfenster anzeigen (*kap06\alert.html*)

```
<script>
  alert("Herzlich willkommen \nauf meiner Webseite.");
</script>
```



Eingabefenster

In einem Eingabefenster kann der Benutzer einen Text eingeben. Die Seite wird versendet und dann im eigenen Browser mittels JavaScript ausgewertet.

<pre>prompt(Text, Vorgabe); prompt("Eingabe", ["Vorgabe"]);</pre>	<p>Die Funktion <code>prompt()</code> öffnet das Eingabefenster. Geben Sie in Anführungszeichen den Text ein, der angezeigt werden soll. Optional geben Sie anschließend eine Vorgabe für das Eingabefeld an. Da von der Funktion ein Wert zurückgegeben wird, muss eine Variable angegeben werden, die diesen Wert speichern soll. Die Rückgabewerte können sein:</p> <ul style="list-style-type: none"> ✓ Benutzereingabe, die der Benutzer mit OK bestätigt ✓ Wert <code>null</code>, wenn die Eingabe abgebrochen wurde
---	---

Beispiel: Benutzereingabe ausgeben (*kap06\prompt.html*)

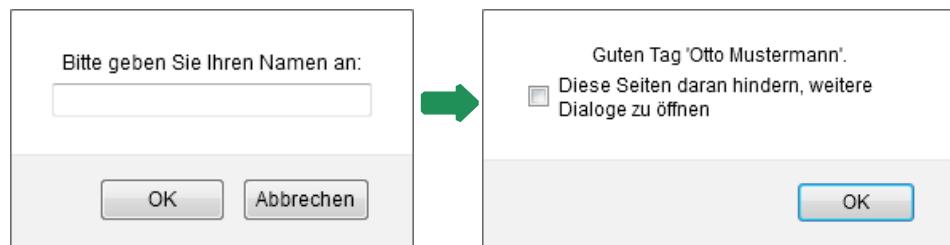
Der Besucher soll im Eingabefenster seinen Namen hinterlassen, der im darauf folgenden Dialogfenster vom Programm ausgegeben wird.

<pre><html> <head> <meta charset="utf-8"> <title>Interaktion</title> </head> <body> <script> ① name = prompt("Bitte geben Sie Ihren Namen an:", ""); ② alert("Guten Tag '" + name + "'."); </script> </body> </html></pre>
--

 In diesem Beispiel wurde die Variable `name` definiert und ihr gleichzeitig ein Wert zugewiesen, um den Code kurz zu halten. In der Praxis ist es sicherer, wenn Sie – wie in Kapitel 5 gezeigt – Ihre Variablen schon vor ihrer Verwendung definieren und initialisieren werden.

- ① Der Eingabewert des Benutzers wird der Variablen `name` übergeben.
- ② Mit `alert()` wird ein Dialogfenster geöffnet. Als Wert geben Sie eine Zeichenkette und den Variablenwert `name` an.

 Das Dialogfenster gibt entweder den Text mit dem eingegebenen Namen aus oder den Wert `null`, wenn der Besucher die Eingabe abbricht.



Eingabeaufforderung und Ausgabe des Namens in Firefox

Bestätigungsfenster

Ein weiteres Dialogfenster können Sie aufrufen, wenn Sie eine Frage stellen möchten, die vom Anwender nur mit *JA* oder *NEIN* beantwortet werden soll.

<code>confirm("Text");</code>	<p>Die Funktion <code>confirm</code> (<i>Bestätigung</i>) öffnet ein Bestätigungsfenster. Als Text geben Sie die Frage an, die angezeigt werden soll.</p> <p>Bei der Frage, ob der Benutzer auf <i>OK</i> oder <i>Abbrechen</i> geklickt hat, muss eine Variable angegeben werden, die diesen Wert speichert. Die Rückgabewerte können sein:</p> <ul style="list-style-type: none"> ✓ <code>true</code>, wenn der Benutzer die Frage mit <i>OK</i> bestätigt hat ✓ <code>false</code>, wenn er die Frage verneint, also die Schaltfläche <i>Abbrechen</i> betätigt
-------------------------------	--

Beispiel: Ja-/Nein-Frage (kap06\confirm.html)

```
<script type="text/javascript">
    wahl = confirm("Sind Sie das erste Mal hier?");
</script>
```

Sind Sie das erste Mal hier?

OK Abbrechen

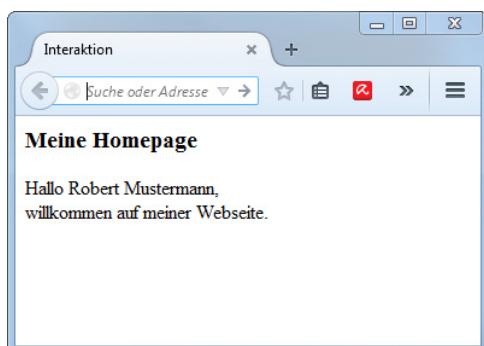
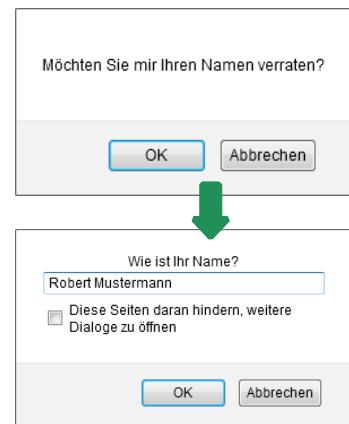
Komplettbeispiel: Interaktionsfenster (kap06\eingaben.html)

Das Beispiel wird die soeben erläuterten Interaktionsfenster beinhalten. Dabei soll der Name des Besuchers erfragt und auf der Webseite ausgegeben werden. Gibt der Besucher keinen Namen ein, soll ihm automatisch der Name "Frau/Herr Namenlos" zugewiesen werden.

```
<html>
<head>
<meta charset="utf-8">
<title>Interaktion</title>
</head>
<body>
<h3>Meine Homepage</h3>
① <script>
②   var name = "Frau/Herr Namenlos";
③   nameVerraten = confirm("Möchten Sie mir Ihren Namen verraten?");
④   if (nameVerraten == true) {
⑤     name = prompt("Wie ist Ihr Name?", "");
⑥   if (name == "" || name == null) {
⑦     alert("Name unbekannt.");
⑧     name = "";
⑨   }
⑩   document.write("Hallo " + name + ",<br>willkommen auf meiner Webseite.");
</script>
</body>
</html>
```

- ① Das Skript wird in diesem Fall innerhalb des Dokumentskörpers geschrieben, da der ausgegebene Text nach der Überschrift (`h3`) erscheinen soll.
- ② Die Variable `name` wird mit "Frau/Herr Namenlos" initialisiert und soll später den Namen des Benutzers speichern.

- ③ Es öffnet sich das Abfragefenster, in dem der Besucher entscheiden kann, ob er seinen Namen verraten möchte. Die Variable `nameVerraten` erhält das Ergebnis `true` beim Bestätigen mit `Ok` oder `false` beim Klick auf `ABBRECHEN`.
- ④ Hier erfolgt das Auslesen der Variablen `nameVerraten`. Ist diese `true`, d. h., möchte der Benutzer seinen Namen angeben, wird die Zeile ⑤ abgearbeitet.
- ⑤ Das Eingabefenster wird geöffnet. Der Variablen `name` wird der eingegebene Name des Besuchers übergeben.
- ⑥ Es folgt die Abfrage der Variablen `name`. Wurde kein Name eingegeben (`name = " "`) oder (`||`) hat der Besucher abgebrochen (`name = null`), werden die nächsten beiden Zeilen betrachtet. Trifft keiner der beiden Vergleiche zu, werden die Zeilen ⑦ und ⑧ nicht ausgeführt.
- ⑦ Das Dialogfenster wird mit "Name unbekannt" eingeblendet.
- ⑧ Der Variablen `name` wird eine leere Zeichenkette zugewiesen.
- ⑨ Mit der Funktion `document.write` wird der in Klammern stehende Text in das aktive HTML-Dokument eingefügt. Dabei können ein beliebiger Text und HTML-Tags angegeben werden. Um den Namen des Besuchers auszugeben, wird die entsprechende Variable mit dem Verknüpfungsoperator `(+)` in den Text eingefügt.
- ⑩ Das JavaScript-Skript wird beendet.



Resultat nach Eingabe des Namens



Resultat, wenn der Name nicht angegeben wird

7 Objektmodell in JavaScript

In diesem Kapitel erfahren Sie

- ✓ wie das Objektmodell zum Zugriff auf den Browser aufgebaut ist
- ✓ wie Fenster als Objekte angesprochen werden
- ✓ wie Sie Browser-Daten, das Datum und die Uhrzeit auslesen
- ✓ nützliche Beispiele für den Gebrauch im Internet

Voraussetzungen

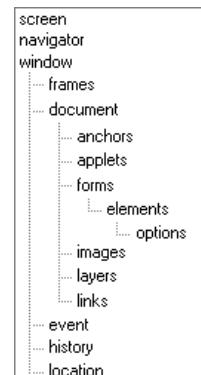
- ✓ Fortgeschrittene HTML-Kenntnisse

7.1 Einführung in das Objektmodell

Objekte sind Datenelemente mit Eigenschaften und oft auch mit objektgebundenen Methoden). In JavaScript wird Ihnen ein Objektmodell angeboten, um beispielsweise auf ein Element eines Formulars zugreifen und dessen Inhalt oder Wert ändern zu können.

Das Konzept des objektorientierten Programmierens hat sich wegen seiner Vorteile schon lange durchgesetzt. Es bildet reale und abstrakte Objekte auf Codestrukturen ab, die leichter verständlich sind und zusammengehörende Daten und Funktionen kapseln. Objekte bieten dem Programmierer wiederverwendbaren Code und erlauben eine bessere Lokalisierung von Fehlern.

Objekte sind nichts anderes als Listen für logisch zusammengehörende Variablen und Funktionen, die als Eigenschaften und Methoden des Objekts bezeichnet werden. Die Eigenschaften sind eine Ansammlung von Werten, die das Objekt näher beschreiben. Methoden sind objektive Funktionen, welche die Eigenschaften oder andere Werte ändern.



In JavaScript gibt es bereits vordefinierte Objekte, die häufig benötigte Methoden beinhalten. Mathematische Methoden sind in diesen Objekten genauso vorhanden wie Methoden zur Berechnung und Verwendung von Uhrzeiten oder Kalendertagen.

Das Objektmodell ist in einer Hierarchie geordnet. Die hierarchiehöchsten Objekte sind das Bildschirmobjekt `screen`, das Fensterobjekt `window` und das Navigator-Objekt `navigator`.

Fenster haben Eigenschaften wie einen Titel, eine Fenstergröße usw. Der Inhalt eines Fensters, also das HTML-Dokument, ist das nächstniedrigere Objekt `document`. Dieses HTML-Dokument enthält bestimmte Elemente, wie z. B. Grafiken, Verweise, Formulare usw. Für jedes dieser Elemente gibt es Objekte, beispielsweise das Objekt `forms` für Formulare. Um auf die Elemente des Formulars (Eingabefelder, Auswahllisten oder Schaltflächen) zugreifen zu können, existiert das Unterobjekt `elements`, mit dem Sie einzelne Felder und andere Elemente innerhalb eines Formulars ansprechen können.

Die beiden Objekte `navigator` und `screen` sind unabhängig von anderen Objekten und geben Auskunft über den Browser sowie Informationen über den Bildschirm des Anwenders.

Neben den hierarchisch geordneten JavaScript-Objekten gibt es auch solche, die nicht direkt in die Hierarchie passen. Das sind zum Beispiel Objekte für Datums- und Zeitrechnung, für mathematische Aufgaben oder für Zeichenkettenverarbeitung.

- ✓ Manipulation von Zeichenketten (Objekt String)
- ✓ Mathematische Funktionen (Objekt Math)
- ✓ Zeit- und Datumsfunktionen (Objekt Date)
- ✓ Behandlung von Datentypfeldern (Objekt Array)

Jedes einzelne Objekt enthält Eigenschaften, die Sie auslesen können, z. B. die Adresse der geladenen Webseite oder den Namen des Fensters. Außerdem erlauben Ihnen die Objektmethoden das Ändern der Objekteigenschaften. So können Sie beispielsweise den Inhalt eines Eingabefeldes ändern.

Aufgrund der Menge an Objekten mit deren Eigenschaften und -methoden werden Ihnen anhand von ausgesuchten Beispielen nachfolgend die wichtigsten Objekte sowie deren Eigenschaften und Methoden nähergebracht. Für weitergehende Informationen zu allen JavaScript-Objekten und deren Funktionen finden Sie im Anhang dieses Buches entsprechende Referenzen.

Ein wichtiges Hilfsmittel zur Fehlersuche („Debuggen“) in Webseiten mit JavaScript ist die Browserkonsole, die Sie in allen modernen Browsern mit der Taste **F12** aufrufen können.



7.2 Objekt navigator

Um die Eigenschaften eines Browsers auslesen zu können, wird das Objekt `navigator` angewendet. Es besitzt acht Eigenschaften und eine Methode.

Eigenschaften	<code>appCodeName, appName, appVersion, language, mimetypes, platform, plugins</code> (Firefox und Opera), <code>userAgent</code>
Methoden	<code>javaEnabled</code>

Die einzelnen Eigenschaften des Objektmodells werden von jedem Browser unterschiedlich interpretiert. Deshalb hat es sich bei den Autoren von Webseiten eingebürgert, für jeden Browser einen speziell angepassten JavaScript-Code zu entwickeln. Um herauszubekommen, welchen Browser der Besucher der Webseite benutzt, wird die Eigenschaft `appName` eingesetzt. Diese gibt den Namen des Browsers an das Programm zurück. Eine weitere Spezifizierung kann mit der Nutzung von `appVersion` durchgeführt werden, da diese Eigenschaft die spezielle Versionsnummer des Browsers zurückliefert.

Beispiel: Angaben über verwendeten Browser anzeigen ([kap07\objekt_navigator.html](#))

Dieses Beispiel liefert einige Angaben über Ihren verwendeten Browser, wie z. B. den Namen, die Version sowie dessen interne Bezeichnung.

```
<html>
<head>
    <meta charset="utf-8">
    <title>Objekt: navigator</title>
    <style type="text/css">
        body { font: small "Courier New", Courier, Arial; }
    </style>
</head>
<body>
    <h2>Eigenschaften des navigator-Objekts</h2>
    <script>
        document.write("<p><b>Codename des Browsers mit navigator.appCodeName:</b><br>" +
                     + navigator.appCodeName + "<br>");
        document.write("<p><b>Name des Browsers mit navigator.appName:</b><br>"
```

```

        + navigator.appName + "<br>");  

document.write("<p><b>Version des Browsers mit navigator.appVersion:</b><br>"  

        + navigator.appVersion + "<br>");  

document.write("<p><b>Bezeichnung des Browsers mit navigator.userAgent:</b><br>"  

        + navigator.userAgent + "<br>");  

document.write("<p><b>Java-Applets möglich? mit navigator.javaEnabled():</b><br>"  

        + navigator.javaEnabled() + "<br>");  

document.write("<p><b>Plattform mit navigator.platform:</b><br>"  

        + navigator.platform + "<br>");  

for (var i=0; i<navigator.plugins.length; i++) {  

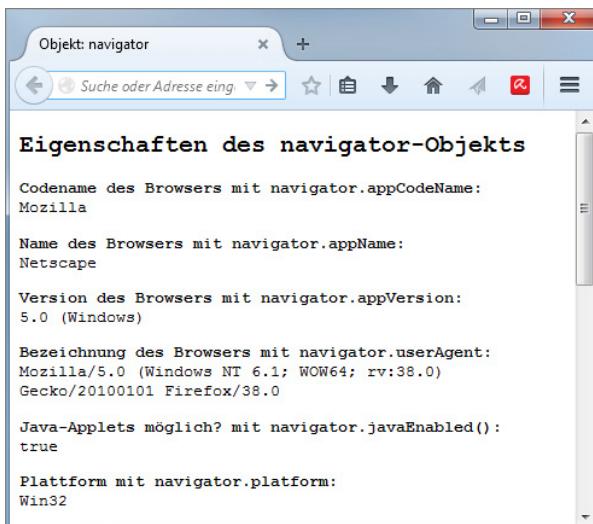
    document.write("<p><b>Installierte PlugIns mit navigator.plugins[]:</b><br>"  

        + navigator.plugins[i].name + "<br>");  

}
</script>
</body>
</html>

```

Ergebnis



Eigenschaften von Firefox

Die Informationen zum verwendeten Browser sind dann wichtig, wenn Verzweigungen für browserspezifische JavaScript-Anweisungen benötigt werden. Einige Browser weisen auch in den neuen Versionen Unterschiede in ihren Implementierungen von JavaScript auf. Somit ist das Verzweigen in unterschiedliche Codesegmente nötig, um Fehlermeldungen in dem einen oder anderen Browser zu vermeiden.

Beispiel: Browserweiche (kap07\ navigator_weiche.html)

Ein typischer Einsatzzweck für das Objekt `navigator` ist die sogenannte Browserweiche, in der die verwendeten Browser abgefragt und speziell für die verschiedenen Browser angepasste Funktionen ausgeführt werden.

 Da die beiden Browser Google Chrome und Apple Safari wie der Mozilla Firefox auf der Browserroutine (Engine) Netscape aufbauen, müssen Sie zur speziellen Unterscheidung die Bezeichnung des jeweiligen Browsers mithilfe der Eigenschaft `navigator.userAgent` überprüfen.

```

<script>
if (parseInt(navigator.appVersion) >= 4) {
    if(navigator.appName == "Netscape") {           // Firefox, Chrome, Safari
        if (navigator.userAgent.indexOf("Chrome") != -1) {
            alert("Sie verwenden Google Chrome.");
        } else if (navigator.userAgent.indexOf("Safari") != -1) {

```

```

        alert("Sie verwenden Apple Safari.");
    } else {
        alert("Sie verwenden Mozilla Firefox.");
    }
} else if(navigator.appName.indexOf("Internet Explorer") != -1) {
    alert("Sie verwenden den Internet Explorer.");
} else if(navigator.appName == "Opera") {
    alert("Sie verwenden den Opera-Browser.");
} else {
    alert("Sie verwenden einen anderen Browser.");
}
} else {
    alert("Sie verwenden einen älteren Browser.");
}
</script>

```

7.3 Objekt window

Das Objekt **window** (*Fenster*) steht in der Objekthierarchie an oberster Stelle. Alle anderen Objekte, die das im Fenster geladene Dokument beschreiben, sind Eigenschaften des **window**-Objekts. Es erlaubt Abfragen zu den einzelnen Dokumentfenstern und ermöglicht es, neue Fenster zu öffnen sowie deren Eigenschaften (Name, Höhe, Breite usw.) festzulegen. Auch das Schließen von Fenstern ist hiermit möglich.

Eigenschaften	closed, defaultStatus, name, status, document, frames, history, innerHeight, innerWidth, length, location, locationbar, menubar, navigator, opener, outerHeight, outerWidth, pageXOffset, pageYOffset, parent, personalbar, scrollbars, self, status, statusbar, toolbar, top, window
Methoden	alert(), back(), blur(), captureEvents(), clearInterval(), clearTimeout(), close(), confirm(), find(), focus(), home(), disableExternalCapture(), forward(), handleEvent(), moveBy(), moveTo(), open(), print(), prompt(), releaseEvents(), stop(), resizeBy(), resizeTo(), routeEvent(), scrollBy(), scrollTo(), setInterval(), setTimeOut(), navigate()

Um eine Eigenschaft oder eine Methode aufzurufen, wird ihnen das entsprechende Objekt vorangestellt.

Objekt.Eigenschaft
Objekt.Methode

Dies bedeutet, um beispielsweise das bereits bekannte Dialogfenster `alert` aufzurufen, müssen Sie `window.alert()` angeben. Im vorigen Kapitel haben Sie jedoch gesehen, dass dies auch ohne Angabe des `window`-Objekts funktioniert. Dies ist richtig, denn das oberste Objekt `window` wird, wenn es nicht angegeben wird, automatisch von JavaScript davorgesetzt.

Zeitlicher Funktionsaufruf

Mit der `window`-Methode `setTimeout` haben Sie die Möglichkeit, eine Anweisung oder Funktion nach einer bestimmten Verzögerungszeit auszulösen. Dazu geben Sie innerhalb der Klammern den entsprechenden JavaScript-Code an. Als zweite Angabe folgt die Verzögerungszeit in Millisekunden.

Beispiel

Nach 10 Sekunden (10.000 Millisekunden) soll die Funktion `zeitvorbei` aufgerufen und ausgeführt werden.

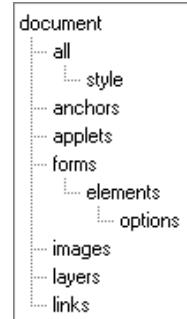
```
<script>
    window.setTimeout('zeitvorbei()',10000);
</script>
```

7.4 Objekt document

Das Objekt `document` kann eine Vielzahl anderer Objekte enthalten. Dies hängt davon ab, welche HTML-Tags auf einer Webseite enthalten sind. So enthält das `document`-Objekt das Unterobjekt `forms`, sobald sich ein Formular `<form>...</form>` auf einer Webseite befindet. Weitere Unterobjekte sind die `anchors`- (Verweisanker), `applets`- (Java-Applets), `images`- (Grafiken) und `links`-Objekte (Verweise).

In diesem Abschnitt werden Ihnen die wesentlichen Eigenschaften und Methoden erklärt, die sich nicht auf ein bestimmtes HTML-Tag beziehen.

Die einzelnen Objekte eines jeden Elements werden im nachfolgenden Kapitel näher erläutert.



Eigenschaften

Diese Tabelle gibt Ihnen einen Überblick über die Eigenschaften und Methoden des `document`-Objekts.

Eigenschaften	<code>alinkColor</code> , <code>bgColor</code> , <code>cookie</code> , <code>fgColor</code> , <code>lastModified</code> , <code>linkColor</code> , <code>location</code> , <code>referrer</code> , <code>title</code> , <code>URL</code> , <code>vlinkColor</code>
Methoden	<code>clear()</code> , <code>close()</code> , <code>open()</code> , <code>write()</code> , <code>writeln()</code> , <code>getElementById()</code> , <code>getElementsByName()</code> , <code>getElementsByTagName()</code>

Die meisten der Eigenschaften entsprechen den Tags in HTML. Mit diesen Eigenschaften können die Attribute der Tags einer Webseite ausgelesen werden. Im Quelltext wurde das oberste Objekt `window` nicht angegeben, da dieses vom JavaScript-Interpreter der Browser standardmäßig vorangesetzt wird.

```
<body>
<h2>Eigenschaften des document-Objekts</h2>
<script>
    document.write("<b>Titel der Webseite:</b> " + document.title + "<br>");
    document.write("<b>Adresse der Webseite:</b> " + document.location +
    "<br>") ;
    document.write("<b>Adresse der Webseite:</b> " + document.URL + "<br>") ;
    document.write("<b>Farbe des Hintergrunds:</b> " + document.bgColor +
    "<br>") ;
    document.write("<b>Farbe der Schrift:</b> " + document.fgColor +
    "<br>") ;
    /* usw. */
</script>
</body>
```

Inhalte in ein Dokument einfügen

```
window.document.write();
window.document.writeln();
```

Die Methoden, die es erlauben, in ein Dokument zu schreiben, dürften Ihnen bekannt vorkommen, denn sie wurden in diesem Buch schon häufig verwendet. Die beiden Methoden sind `write()` und `writeln()`. Sie schreiben die in Klammern stehenden Zeichenketten und/oder Variablen in das aktuelle Dokument. Der einzige Unterschied besteht darin, dass `write` (*schreiben*) alle Ausdrücke ohne Zeilenumbrüche in das Dokument schreibt. Die Methode `writeln` (*write line = in Zeile schreiben*) fügt einen Zeilenumbruch ein. Dieser Umbruch ist nur im Quelltext sichtbar und entspricht nicht dem HTML-Tag `
`.

HTML-Tags als Objekte von `document`

Da der Zugriff auf die Elemente der Dokumentenhierarchie vom HTML-Quelltext abhängt, werden die Objekte der Elemente mit ihrem Namen hinterlegt und zusätzlich in Feldern angelegt.

- ✓ `window.document.images []` enthält alle Grafiken in der Reihenfolge, in der die ``-Tags im HTML-Quelltext verwendet werden.
- ✓ `window.document.anchors []` enthält alle Anker entsprechend den Tags ``.
- ✓ `window.document.links []` enthält alle Links entsprechend den Tags ``.
- ✓ `window.document.forms []` enthält alle Formulare, die durch `<form>`-Tags definiert wurden.

7.5 Objekt `event`

Das `event`-Objekt (*event = Ereignis*) befindet sich seit der JavaScript-Version 1.2 in der Hierarchie unter dem Objekt `window`. Mit diesem Objekt können Sie auf Ereignisse wie Mausklicks oder Tastatureingaben reagieren und JavaScript-Code ausführen. Beispielsweise können Sie bei einem Mausklick ermitteln, ob die linke oder rechte Maustaste gedrückt wurde, oder bei einem Tastendruck die betätigte Taste ermitteln.

Anwenderereignisse können Sie überwachen, indem Sie in einem HTML-Tag eine Ereignisabfrage hinzufügen (`onClick`, `onMouseOver`, `onKeyPress` usw.).

Die Browser ermöglichen die Abfrage der betätigten (Maus-)Tasten. Allerdings stellt der Internet Explorer andere Konstanten zur Verfügung als die beiden Browser Firefox und Opera. So sind Sie gezwungen, die entsprechenden Anweisungen für jeden Browser separat anzugeben.



Die folgende Tabelle listet die unterschiedlichen Eigenschaften des `event`-Objekts für die Browser Internet Explorer, Firefox und Opera auf. Gleichbedeutende Eigenschaften sind auf einer Zeile gegenübergestellt. Das heißt, dass beispielsweise die Explorer-Eigenschaften `altKey`, `ctrlKey`, `shiftKey` der Eigenschaft `modifiers` im Firefox bzw. in Opera entsprechen.

Eigenschaften	Für den Internet Explorer:	Für Firefox und Opera:
	Für den Internet Explorer: <code>altKey</code> , <code>ctrlKey</code> , <code>shiftKey</code> <code>clientX</code> , <code>clientY</code> <code>keyCode</code> <code>offsetX</code> , <code>offsetY</code> <code>x</code> , <code>y</code> <code>button</code>	Für Firefox und Opera: <code>modifiers</code> <code>screenX</code> , <code>screenY</code> <code>which</code> , <code>type</code> <code>layerX</code> , <code>layerY</code> <code>pageX</code> , <code>pageY</code>

Firefox und Opera

Events können von window-, frame- und document-Objekten abgefangen werden. Sie legen die Funktion fest, die bei einem bestimmten Event ausgeführt werden soll, und weisen sie der entsprechenden Objekt-Methode zu.

Beispiel

Es wird der Klick mit der linken Maustaste in einen beliebigen Fensterbereich des Browsers abgefangen.

```
<script>
    function klick()
    {
        alert('Dies war ein einfacher Mausklick!');
    }
    window.onclick=klick;
</script>
```

Damit werden alle click-Ereignisse an die Funktion klick() übergeben. In der Funktion kann das Ereignis verarbeitet oder an das eigentliche Objekt, das angeklickt wurde, weitergegeben werden.



Standardmäßig können Sie im Browser Opera nicht die rechte Maustaste abfragen. Dies muss explizit in den JavaScript-Einstellungen zugelassen werden.

Internet Explorer

Der Internet Explorer benötigt keine spezielle Methode, um Ereignisse abzufangen. Jedes Ereignis wird entlang der Objekthierarchie abgeglichen. Das heißt beispielsweise, dass ein click-Ereignis auf einer Formular-Schaltfläche nach seinem Event-Handler automatisch an den onClick-Handler des Formulars und dann an den des Dokuments usw. weitergegeben wird. Praktisch jedes HTML-Element kann somit mit einem Event-Handler ausgestattet werden.

Das event-Objekt existiert im Internet Explorer nicht. Die Events sind im Internet Explorer eine Eigenschaft des window-Objekts. Die Eigenschaften von window.event werden bei jedem Ereignis überschrieben.

Um z. B. das automatische Weitergeben des Ereignisses zu stoppen, muss der Event-Handler die Eigenschaft cancelBubble auf den Wahrheitswert true setzen:

```
<input type="button" onClick="klick(); window.event.cancelBubble=true">
```

Dadurch wird das Weiterleiten des Ereignisses beendet, und der Handler der Schaltfläche war der letzte, der durch das Ereignis aufgerufen wurde.

Beispiel: ASCII-Codes betätigter Tasten ausgeben ([kap07\ objekt_event-taste.html](#))

In einem mehrzeiligen Eingabefeld werden die Tastatureingaben abgefangen, und der entsprechende ASCII-Code wird in der Statuszeile des Browsers ausgegeben.

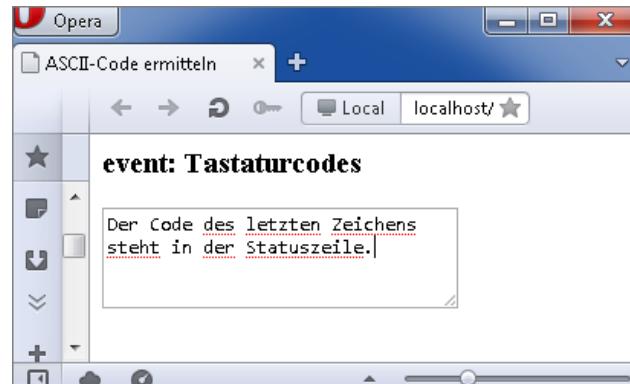
```
<html>
<head>
    <meta charset="utf-8">
    <title>ASCII-Code ermitteln</title>
    <script>
        var isNav, isIE = false;
```

```

②  if (parseInt(navigator.appVersion) >= 4) {
    if ((navigator.appName == "Netscape") || (navigator.appName == "Opera")){
        isNav = true;
    } else {
        isIE = true;
    }
}
③ function ZeigeTaste(evt)
{
    if (isNav) {
        window.status = evt.which;
    } else {
        window.status = window.event.keyCode;
    }
    return false;
}
</script>
</head>
<body>
    <h3>event: Tastaturcodes</h3>
    <form>
        <textarea cols="30" rows="4" onKeyPress="ZeigeTaste(event)"></textarea>
    </form>
</body>
</html>

```

- ① Zuerst wird unterschieden, ob ein Browser mit einer Versionsnummer höher oder gleich 4 verwendet wird.
- ② Je nachdem, welcher Browser verwendet wird, erhält eine der Variablen `isNav` bzw. `isIE` den Wahrheitswert `true`.
- ③ Die Funktion `ZeigeTaste (evt)` wird später vom `onKeyPress`-Handler des Eingabefeldes aufgerufen.
- ④ Ist der Browser Opera oder ein auf Netscape basierender Browser (`isNav`), erhalten Sie den ASCII-Code aus der `which`-Eigenschaft des übergebenen Event-Objekts. Ansonsten steht der Wert in der Eigenschaft `window.event.keyCode`. In beiden Fällen wird der ASCII-Code in der Statuszeile ausgegeben.
- ⑤ Im Rumpfbereich der Webseite enthält das Formular das notwendige Eingabefeld. Diesem wird über den Event-Handler `onKeyPress` der Aufruf der Funktion `ZeigeTaste ()` zugewiesen.



Maustasten überwachen

Jeder Anwender kann den Quelltext und die Grafiken einer Webseite auf seinem Computer speichern (vgl. nachfolgende Abbildung). Sie können dies verhindern, vorausgesetzt, der Anwender erlaubt die Ausführung von JavaScript. Sie fangen dazu das Betätigen der Maustasten ab.

Beispiel: Maustasten überwachen ([kap07\objekt_event-maus.html](#))

Zum Kennenlernen der `event`-Eigenschaften entwickeln Sie ein Skript, das in den Browsern das Betätigen der Maustasten überwacht.

```

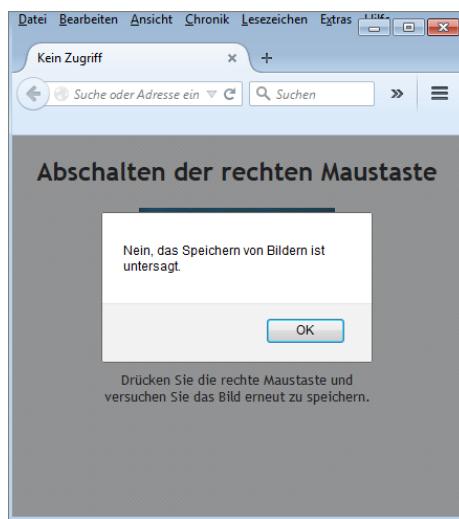
<script>
    var message_lm = "Ja, die linke Maustaste funktioniert!";
    var message_rm = "Nein, das Speichern von Bildern ist untersagt.";
    function wache(e) {
        if (window.navigator.appName == "Microsoft Internet Explorer") {
            if(window.event.button == 2) { // rechte Maustaste
                alert(message_rm);
                return false;
            }
            if(window.event.button == 1) // linke Maustaste
                alert(message_lm);
                return true;
        }
        if ((window.navigator.appName == "Netscape") ||
            (window.navigator.appName == "Opera")) {
            if(e.which == 3) { // rechte Maustaste
                alert(message_rm);
                return false;
            }
            if(e.which == 1) // linke Maustaste
                alert(message_lm);
                return true;
        }
    }
    document.onmousedown = wache;
</script>

```

- ① Die Funktion `wache` wird initialisiert und übernimmt die Informationen zum eingetretenen Ereignis durch das Objekt `e`. Dieses Objekt ist im Beispiel für die Ereignisabfrage in den Browsern Firefox und Opera wichtig.
- ② Der Name des Browsers wird ausgelesen, und es wird verglichen, ob es sich um den Internet Explorer handelt.
- ③ In dem Anweisungsblock wird über die Eigenschaft `button` des `event`-Objekts die betätigte Maustaste abgefragt. Der Wert 2 steht hierbei für die rechte Maustaste. Wurde diese betätigt, erscheint die entsprechende Meldung. Über den Rückgabewert `false` verlassen Sie die Funktion und schalten im Internet Explorer auch die Anzeige des Kontextmenüs ab.
- ④ Die linke Maustaste fragen Sie über den Vergleich mit dem Wert 1 ab und geben eine andere Meldung aus. Die Funktion wird über den Rückgabewert `true` verlassen.
- ⑤ In dieser Zeile überprüfen Sie, ob es sich um den Firefox oder um Opera handelt.
- ⑥ Die Abfrage des Ereignisses erfolgt in den beiden Browsern über das übergebene Objekt `e`. Die entsprechende Eigenschaft lautet `which`. Der Wert 3 steht dabei für die rechte Maustaste.
- ⑦ Die linke Maustaste wird über den Wert 1 identifiziert, und es wird eine Meldung ausgegeben.
- ⑧ Diese Anweisung initialisiert die Überwachung der Tastatureingabe. Tritt das Ereignis `onmousedown` (*wenn Maustaste gedrückt wird*) ein, wird die Funktion `wache()` mit der Übergabe der Ereignisabfrage aufgerufen und abgearbeitet.



Kontextmenü zum Speichern von Grafiken



Gesperrtes Kontextmenü in Firefox

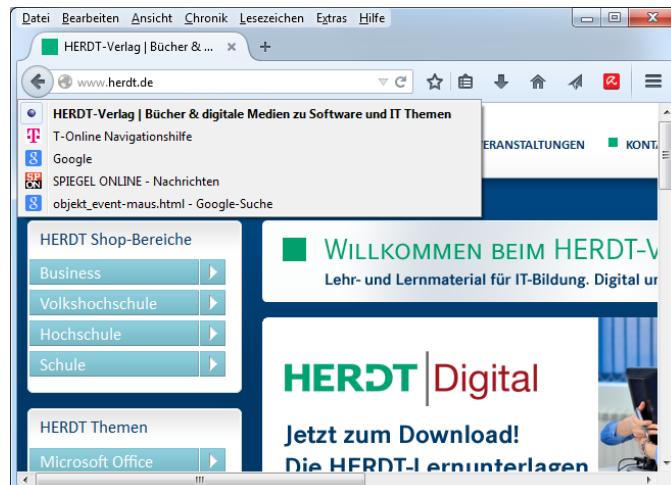
Das Abschalten der rechten Maustaste verhindert jedoch nicht das Speichern der Grafik, da sich diese auch im Cache des Browsers auf der Festplatte finden lässt bzw. das Überwachen der Maustasten umgangen werden kann, wenn JavaScript deaktiviert wird.



7.6 Objekt `history`

Das `history`-Objekt (*history = Geschichte*) befindet sich in der Hierarchie unter dem Objekt `window`. Laden Sie eine Webseite in Ihren Browser, wird ihre URL im Verlauf des Browsers gespeichert (im Firefox als Chronik bezeichnet). Dies können Sie überprüfen, indem Sie mit der Maus auf den nach unten weisenden Pfeil neben den Navigationsschaltflächen des Browsers klicken. Es wird eine Liste der zuletzt besuchten Webseiten angezeigt.

Mit diesem Objekt erhalten Sie einen bedingten Zugriff auf diese gespeicherten URLs. Bedingt heißt, dass Sie nur vorwärts und rückwärts auf die Einträge zugreifen können, jedoch nicht direkt. Dies dient zum Schutz der Privatsphäre des Nutzers.



Auflistung der zuletzt besuchten Seiten

Eigenschaften	<code>length</code>
Methoden	<code>back()</code> , <code>forward()</code> , <code>go()</code>

Die einzige Eigenschaft `length` liefert Ihnen die Anzahl der Einträge im Verlauf der aktuellen Sitzung zurück. Die Methode `back` (*zurück*) erlaubt ein schrittweises Zurückblättern und `forward` (*vorwärts*) ein schrittweises Vorwärtsblättern.

Wird `back()` von dem ersten und `forward()` von dem letzten Eintrag in der History aufgerufen, haben diese Methoden keine Auswirkung.



Bei der Methode `go()` können Sie eine bestimmte Anzahl an Seiten überspringen. Eine negative Zahlenangabe (`go(-2)`) lädt eine der vorherigen Seiten, eine positive Angabe (`go(2)`) lädt eine der nachfolgenden Seiten der History.

```
<p>&lt; <a href="javascript:history.back()">Eine Seite zurückblättern</a>
<p>&gt; <a href="javascript:history.forward()">Eine Seite vorwärtsblättern</a>
<p>&lt;&lt; <a href="javascript:history.go(-2)">Zwei Seiten zurückblättern</a>
<p>&gt;&gt; <a href="javascript:history.go(2)">Zwei Seiten vorwärtsblättern</a>
```

7.7 Objekt `location`

Das `location`-Objekt (*Ort*) befindet sich in der Hierarchie unter dem Objekt `window`. Mit diesem Objekt können Sie auf die Adresse des aktuell dargestellten Dokuments zugreifen. Sie können dabei die gesamte URL oder auch einzelne Teile davon auslesen und verarbeiten.

Die nachfolgende Tabelle gibt Auskunft über die Eigenschaften und Methoden dieses Objekts.

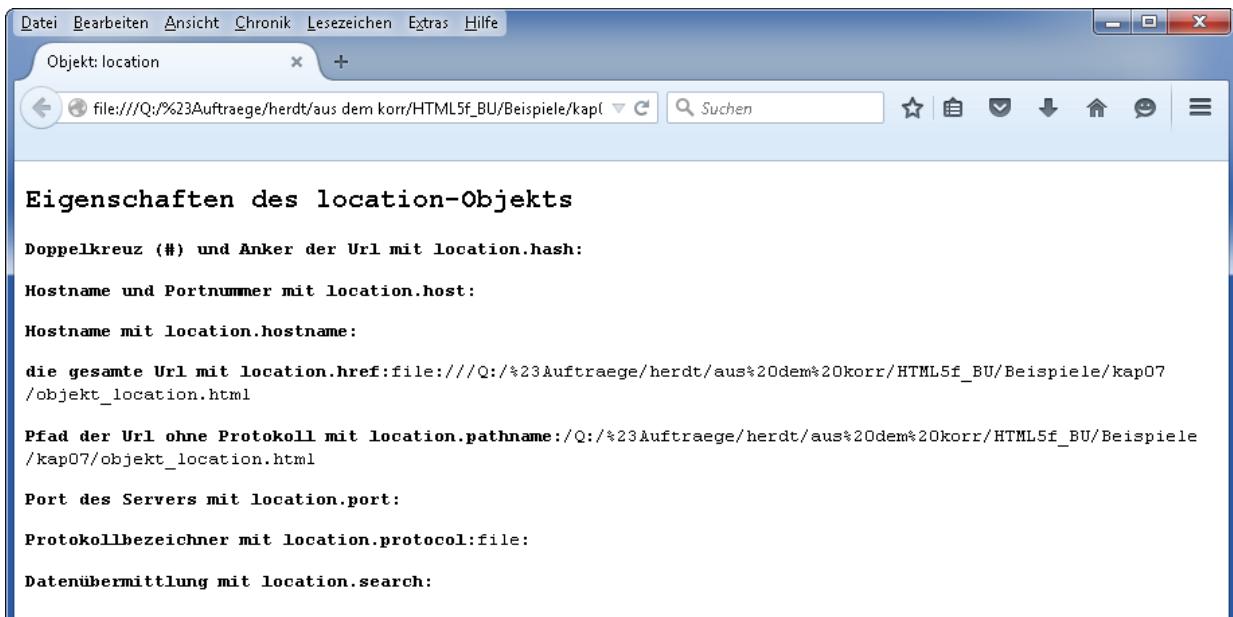
Eigenschaften	<code>hash, host, hostname, href, pathname, port, protocol, search</code>
Methoden	<code>reload, replace</code>

Die Eigenschaften entsprechen im Wesentlichen den Bestandteilen der URL. Die Methode `reload` aktualisiert das Dokument, indem sie es erneut in den Browser lädt.

```
<script>
document.write("<p><b>Doppelkreuz (#) und Anker der Url mit location.hash:</b>" +
    + location.hash + "<br>");
document.write("<p><b>Hostname und Portnummer mit location.host:</b>" +
    + location.host + "<br>");
document.write("<p><b>Hostname mit location.hostname:</b>" +
    + location.hostname + "<br>");
document.write("<p><b>die gesamte Url mit location.href:</b>" +
    + location.href + "<br>");
document.write("<p><b>Pfad der Url ohne Protokoll mit location.pathname:</b>" +
    + location.pathname + "<br>");
document.write("<p><b>Port des Servers mit location.port:</b>" +
    + location.port + "<br>");
document.write("<p><b>Protokollbezeichner mit location.protocol:</b>" +
    + location.protocol + "<br>");
document.write("<p><b>Datenübermittlung mit location.search:</b>" +
    + location.search + "<br>");

</script>
```

Folgende Eigenschaften können aus der in der Adresszeile des Browsers befindlichen Adresse ausgelesen werden:



Auslesen von Eigenschaften einer Webseite (kap09\objekt_location.html)

7.8 Vordefinierte Objekte in JavaScript

Objekt **String** für Zeichenketten

Das Objekt **String** (*Zeichenkette*) wird mit dem Schlüsselwort `new` erzeugt. Als Parameter wird mit `String()` die entsprechende Zeichenkette übergeben.

Objekterzeugung	<code>Variable = new String("Zeichenkette");</code>
-----------------	---

Ein Objekt vom Typ **String** besitzt eine Eigenschaft und weitere Methoden, um Zeichenketten zu verarbeiten und auszugeben.

Eigenschaften	<code>length</code>
Methoden	<p>Verarbeitung</p> <p><code>charAt()</code>, <code>charCodeAt()</code>, <code>concat()</code>, <code>indexOf()</code>, <code>lastIndexOf()</code>, <code>slice()</code>, <code>split()</code>, <code>substr()</code>, <code>substring()</code>, <code>toLowerCase()</code>, <code>toUpperCase()</code></p> <p>Ausgabe in HTML</p> <p><code>anchor()</code>, <code>big()</code>, <code>blink()</code>, <code>bold()</code>, <code>fixed()</code>, <code>fontcolor()</code>, <code>fontsize()</code>, <code>italics()</code>, <code>link()</code>, <code>small()</code>, <code>strike()</code>, <code>sub()</code>, <code>sup()</code></p>

Die Eigenschaft `length` liefert die Anzahl der Zeichen einer Zeichenkette. Mit den Methoden zur Verarbeitung von Strings können Sie bestimmte Stellen der Zeichenketten auslesen und verändern.

Zur Darstellung der verschiedenen Verarbeitungsmöglichkeiten soll als Beispiel die Variable `zitat` mit der Zeichenkette "Sein oder nicht sein," verwendet werden (`zitat = "Sein oder nicht sein,"`). Der Zugriff auf das erste Zeichen erfolgt im Folgenden über den Index 0. Auf das zweite Zeichen greifen Sie mit dem Index 1 zu usw.

Methode	Erläuterung	Ergebnis
<code>zitat.charAt(6);</code>	Mit <code>charAt(x)</code> können Sie das Zeichen auslesen, das sich an der $x+1$ -ten Stelle befindet.	d
<code>zitat.charCodeAt(6);</code>	<code>charCodeAt(x)</code> liefert den ASCII-Code des $x+1$ -ten Zeichens.	100
<code>zitat.concat(" das ist hier die Frage.");</code>	Verbindet die Zeichenkette mit der in Klammern angegebenen Zeichenkette	Sein oder nicht sein, das ist hier die Frage.
<code>zitat.indexOf('c');</code>	Sucht nach dem ersten Auftreten der angegebenen Zeichenkette und gibt zurück, an der wievielten Stelle sie im String gefunden wurde. Wird das Zeichen nicht gefunden, wird -1 zurückgegeben.	12
<code>zitat.lastIndexOf('e');</code>	Die Suche nach dem String startet am Ende der Zeichenkette und verläuft rückwärts.	17
<code>zitat.slice(5, 9);</code>	Extrahiert einen Teil der Zeichenkette. Die Parameter sind die Start- und Endposition in der Zeichenkette. Beim Weglassen des zweiten Parameters wird der Rest der Zeichenkette ausgelesen.	oder
<code>zitat.split(' ');</code>	Diese Methode teilt eine Zeichenkette in mehrere Teile. Die Trennung erfolgt an den Positionen des angegebenen Trennzeichens.	('Sein', 'oder', 'nicht', 'sein.. .')
<code>zitat.substr(5, 4);</code>	Diese Methode funktioniert ähnlich der Methode <code>slice()</code> . Es werden jedoch der Beginn der Teilzeichenkette und deren Länge angegeben.	oder
<code>zitat.substring(5, 9);</code>	Die Funktionsweise entspricht der Funktion der <code>slice()</code> -Methode.	oder
<code>zitat.toLowerCase();</code>	Der Rückgabewert ist die Zeichenkette, deren Buchstaben alle in Kleinbuchstaben umgewandelt wurden.	sein oder nicht sein,
<code>zitat.toUpperCase();</code>	Der Rückgabewert ist die Zeichenkette, deren Buchstaben alle in Großbuchstaben umgewandelt wurden.	SEIN ODER NICHT SEIN,

Beispiel: Textkette verarbeiten (kap07\objekt_string.html)

```

Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
Objekt String
Zeichenketten-Verarbeitung

zitat1 = Sein oder nicht sein,
zitat2 = das ist hier die Frage,
zitat1.length = 21 Zeichen
zitat1.charAt(6) = d
zitat1.charCodeAt(6) = 100
zitat1.concat(zitat2) = Sein oder nicht sein, das ist hier die Frage.
zitat1.indexOf('c') = 12. Stelle
zitat1.lastIndexOf('e') = 17. Stelle
zitat1.slice(5, 9) = oder
zitat1.split(' ') = Sein, oder, nicht, sein,
zitat1.substr(5, 4) = oder
zitat1.substring(5, 9) = oder
zitat1.toLowerCase() = sein oder nicht sein,
zitat1.toUpperCase() = SEIN ODER NICHT SEIN,

```

Ausgabe der String-Eigenschaft und -Methoden im Browser Firefox

Die weiteren Methoden zur Ausgabe einer Zeichenkette entsprechen den Tags in HTML. Deshalb werden die Auswirkungen hier nicht näher erläutert, sondern können in der Übungsdatei *kap09\html_string.html* eingesehen werden.

Objekt Math für mathematische Berechnungen

Bei den Eigenschaften des Objekts Math handelt es sich um oft verwendete mathematische Konstanten, die mit einer hohen Genauigkeit zur weiteren Verwendung in JavaScript ausgelesen werden können. Die Methoden des Objekts Math stellen Ihnen zudem einige mathematische Funktionen zur Verfügung.

Eigenschaften	E, LN2, LN10, LOG2E, LOG10E, PI, SQRT1_2, SQRT2
Methoden	abs(), acos(), asin(), atan(), atan2(), ceil(), cos(), exp(), floor(), log(), max(), min(), pow(), random(), round(), sin(), sqrt(), tan()
Aufruf	Variable = math.Eigenschaft Variable = math.Methode()

Die mathematischen Konstanten kennen Sie aus dem Mathematikunterricht. Neu ist in JavaScript die Eigenschaft `Math.SQRT1_2`, welche die Quadratwurzel aus $\frac{1}{2}$ beinhaltet. `Math.SQRT2` ist die Quadratwurzel aus der Zahl 2.

Die Methoden entsprechen in der Schreibweise denen der Mathematik. Weniger bekannte Methoden sind folgende: `Math.atan2(x, y)` ist der Arcus Tangens für eine Gegenkathete x und eine Ankathete y. Die Methode `Math.ceil(x)` liefert die nächste Ganzzahl von x, `Math.floor(x)` die nächst niedrige Ganzzahl und `Math.pow(x, y)` liefert den Wert von x^y .

Sehr nützlich ist die Methode `random`, die eine Zufallszahl zwischen 0 und 1 als Dezimalzahl zurückliefert. Um daraus eine Ganzzahl zu erzeugen, muss diese Zahl mit dem entsprechenden Faktor multipliziert werden:

```
Zufallszahl=math.random()*100; // Zufallszahl zwischen 0 und 100 zu erzeugen
Zufallszahl=math.random()*1000; // Zufallszahl zwischen 0 und 1000 zu erzeugen
```

Beispiel: Zufallsgenerator (*kap07\objekt_math_random.html*)

Die Methode `random()` wird häufig eingesetzt, um z. B. in Spielen zufällige Ausgangssituationen herbeizuführen. Folgendes Beispiel beinhaltet die Funktion `RolleWuerfel()`, die mit der `random()`-Methode zufällige Zahlen zwischen 1 und 6 erzeugt:

```
<html>
<head>
    <meta charset="utf-8">
    <title>W&uuml;rfel</title>
    <script>
        function RolleWuerfel() {
            ①    var Zufallszahl = Math.random() * 6;
            ②    return (Math.floor(Zufallszahl) + 1);
        }
    </script>
</head>
<body>
    <h3>W&uuml;rfelergebnisse:</h3>
    <script>
        ③    for(i = 0; i < 10; i++)
            document.write(RolleWuerfel() + " - ");
    </script>
```

④ `<p>Neu laden
</body>
</html>`

- ① Die Methode `random()` erzeugt bei jedem Aufruf eine zufällige reelle Zahl zwischen 0 und 0.9999. Um im ersten Schritt den Zahlenbereich 0 bis 5 zu erreichen, muss die Zufallszahl mit dem Wert 6 multipliziert werden. Das Ergebnis wird als Gleitkommazahl in der Variable `Zufallszahl` abgelegt.
- ② Vor der Rückgabe wird der Wert der Variablen `Zufallszahl` mit der Methode `floor()` auf die nächste Ganzzahl abgerundet. Dies ergibt eine ganze Zahl zwischen 0 und 5. Die Addition von 1 ergibt das gewünschte Ergebnis einer ganzen Zahl zwischen 1 und 6.
- ③ Im Rumpf des HTML-Dokuments wird die Funktion `RolleWuerfel()` über eine `for`-Schleife zehnmal ausgeführt und das Ergebnis im Browser ausgegeben.
- ④ Über die Methode `location.reload()` fordern Sie den Browser auf, beim Klick auf den Link das Dokument neu zu laden und somit die Zufallsfunktion erneut aufzurufen.

Objekt Date für Zeitangaben

Mit diesem Objekt haben Sie die Möglichkeit, das Datum und die Uhrzeit in JavaScript zu verarbeiten. Dazu besitzt das Objekt eine Reihe von Methoden zum Lesen und Manipulieren von Datum und Uhrzeit.

Es gibt drei verschiedene Möglichkeiten, ein Datumsobjekt zu initialisieren:

Objekterzeugung	<ol style="list-style-type: none"> ① <code>var Variable = new Date();</code> ② <code>var Variable = new Date("englischer_Monatsname Tag, Jahr Stunde:Minute:Sekunde");</code> ③ <code>var Variable = new Date(Jahr, Monat, Tag, Stunde, Minute, Sekunde);</code>
-----------------	---

Geben Sie bei der Initialisierung keine Parameter an ①, wird das Objekt mit dem aktuellen Datum und der aktuellen Uhrzeit initialisiert. Sie können aber auch der Variablen ein bestimmtes Datum und eine bestimmte Uhrzeit zuordnen. Geben Sie die entsprechenden Daten in Form einer Zeichenkette ② oder in numerischer Form ③ an.

Beispiel

```
var zeit1 = new Date() ; // aktuelles Datum und Uhrzeit wird übergeben
var zeit2 = new Date("March 20, 2009 16:15:00") ; // 20.03.2009 um 16:15 Uhr
var zeit2 = new Date(2009, 2, 20, 16, 15, 00) ; // 20.03.2009 um 16:15 Uhr
```

Bei der Initialisierung in numerischer Form (`zeit2`) ist zu beachten, dass die Monate im Wertebereich von 0 bis 11 liegen (0 = Januar, 1 = Februar, ..., 11 = Dezember). Außerdem können Sie auch nur das Datum setzen und die Parameter für die Uhrzeit weglassen. Diese wird dann automatisch auf 0:00 Uhr gesetzt.

```
var zeit3 = new Date(2009, 2, 20) ; // 20.03.2009 um 00:00 Uhr
```

Das Objekt `Date` besitzt mehrere Methoden, um die Uhrzeit und das Datum auszulesen und zu verändern.

Eigenschaften	-
Methoden	<p>Lesen</p> <pre>getFullYear(), getMonth(), getDate(), getHours(), getMinutes(), getSeconds(), getTime(), getTimezoneOffset(), getDay()</pre> <p>Schreiben</p> <pre>setFullYear(), setMonth(), setDate(), setHours(), setMinutes(), setSeconds(), setTime()</pre> <p>Weitere</p> <pre>toGMTString(), toLocaleString(), Date.parse(), Date.UTC()</pre>

Die Methoden `getFullYear` (Jahr), `getMonth` (Monat), `getDate` (Tag), `getHours` (Stunde), `getMinutes` (Minute), `getSeconds` (Sekunde) lesen die aktuellen Daten aus.

Verwenden Sie nicht mehr die bisherige Methode `getYear` (). Aufgrund der Tatsache, dass diese Methode nicht Jahr-2000-kompatibel ist, liefert sie in den Mozilla-Browsern nur die Differenz zum Jahr 1900 zurück. Die neue Methode `getFullYear` () liefert hingegen in allen Browsern das korrekte Jahr zurück.



Die Methoden `set...` ändern die Zeitdaten. Diese Änderungen sind nur innerhalb des JavaScripts gültig, die Systemzeit Ihres Computers bleibt davon unberührt.

`getTime` und `setTime` lesen bzw. schreiben die verstrichenen Millisekunden seit dem 1. Januar 1970 um 0:00 Uhr. `getTimezoneOffset` liefert als Ergebnis die Differenz zur Greenwichzeit (GMT, Greenwich Mean Time) in Minuten; in Deutschland ist dies -60 Minuten. `getDay` liefert den Wert des aktuellen Wochentags zurück. Dabei ist zu beachten, dass Sonntag = 0, Montag = 1, ..., Samstag = 6 ist.

Neben den lesenden und schreibenden Methoden gibt es im `Date`-Objekt vier weitere Methoden, die eine Umwandlung der verschiedenen Datumsformate ermöglichen. Die Methode `toGMTString` gibt die aktuelle Greenwichzeit und `toLocaleString` gibt das aktuelle Datum und die Uhrzeit als landestypische Zeichenkette zurück.

Mit den Methoden `parse` (Datumszeichenkette) und `UTC` (Jahr, Monat, Tag, Std., Min., Sek.) können Sie Daten in Millisekunden ausgeben lassen (`UTC` = Universal Time Coordinate). Beide Methoden stehen bei selbst definierten `Date`-Objekten nicht zur Verfügung. Um diese Methoden zu verwenden, muss zwingend die Schreibweise `Date.parse` (Parameter) bzw. `Date.UTC` (Parameter) benutzt werden.



Beispiel: Datum und Uhrzeit auslesen ([kap07\objekt_date.html](#))

Das folgende Beispiel enthält einige Methoden zum Auslesen und Manipulieren von Datum und Uhrzeit mithilfe von JavaScript. Weiterhin wird das Datum ermittelt und ausgegeben, das in 150 Tagen erreicht wird.

```
<h3> Die aktuelle Uhrzeit</h3>
<script>
①  var zeit = new Date(); // Variable mit aktuellem Datum initialisieren
②  document.write("Datum, Uhrzeit: " + zeit.toLocaleString() + " Minuten<br>");
  document.write("Jahr: " + zeit.getFullYear() + "; ");
  document.write("Monat: " + zeit.getMonth() + "; ");           // Monate 0...11
  document.write("Tag: " + zeit.getDate() + "; ");
  document.write("Wochentag: " + zeit.getDay() + "; ");
  document.write("Stunden: " + zeit.getHours() + "; ");
  document.write("Minuten: " + zeit.getMinutes() + "; ");
  document.write("Sekunden: " + zeit.getSeconds() + "<br> ");
  document.write("Greenwichzeit: " + zeit.toGMTString() + "<br> ");
  document.write("Lokale Ausgabe: " + zeit.toLocaleString() + "<br> ");
  document.write("Abweichung: " + zeit.getTimezoneOffset() + " Minuten<br>");
```

```

③ var jahr = zeit.getFullYear();
var monat = zeit.getMonth() + 1;
var tag = zeit.getDate();
document.write("Heute ist der " + tag + "." + monat + "." + jahr + "<br>");

④ dann = zeit.getTime() + (150*24*60*60*1000); // 150 Tage in Millisekunden
⑤ zeit.setTime(dann);
⑥ document.write("In 150 Tagen ist " + zeit.toLocaleString() + "<br>");
</script>

```

- ① Die Variable `zeit` wird mit dem aktuellen Datum initialisiert und für die weiteren Berechnungen genutzt.
- ② Es werden über die Methoden die verschiedenen Datums- und Zeitwerte ausgegeben. Zu beachten ist, dass beim Monat der Wert immer um 1 niedriger ist.
- ③ Für die Berechnung eines neuen Datums werden die notwendigen Variablen zum Speichern des Jahres, Monats und Tags erstellt und mit den aktuellen Werten initialisiert. Beachten Sie, dass der Monatswert von 0 bis 11 geht und deshalb um 1 erhöht werden muss.
- ④ Die 150 Tage werden in Millisekunden umgerechnet, da nur damit eine Datumsberechnung durchgeführt werden kann.
- ⑤ Das Datum wird über die Methode `setTime()` um 150 Tage vordatiert.
- ⑥ Über die Methode `toLocaleString()` wird das zukünftige Datum im landestypischen Format ausgegeben.

Die aktuelle Uhrzeit

Datum, Uhrzeit: 19.5.2015 12:28:25 Minuten
Jahr: 2015; Monat: 4; Tag: 19; Wochentag: 2; Stunden: 12; Minuten: 28; Sekunden: 25
Greenwichzeit: Tue, 19 May 2015 10:28:25 GMT
Lokale Ausgabe: 19.5.2015 12:28:25
Abweichung: -120 Minuten
Heute ist der 19.5.2015
In 150 Tagen ist 16.10.2015 12:28:25

Verschiedene Ausgaben des aktuellen Datums und der Uhrzeit



Obwohl die Berechnung korrekt durchgeführt wurde, liegt die Uhrzeit in 150 Tagen eine Stunde hinter der aktuellen Uhrzeit. Dies liegt an der zwischenzeitlichen Umstellung von der Winter- zur Sommerzeit.

Beispiel: Aktuelle Uhrzeit anzeigen (`kap07\objekt_date_settimeout.html`)

In dem Beispiel werden Sie mit der windows-Methode `setTimeout()` ständig die aktuelle Uhrzeit auslesen und anzeigen. Mit der Angabe einer Verzögerungszeit starten Sie das Auslesen zu jeder Sekunde.

```

<html>
<head>
    <meta charset="utf-8">
    <title>Die Uhrzeit</title>
    <script>
        ① function Zeitauslese() {
            var jetzt = new Date()
            var stunde = jetzt.getHours()
            var minute = jetzt.getMinutes()
            var sekunde = jetzt.getSeconds()
            var zeit=''

            zeit += ((stunde < 10) ? "0" : "") + stunde
            zeit += ((minute < 10) ? ":0" : ":") + minute
            zeit += ((sekunde < 10) ? ":0" : ".") + sekunde
            document.forms[0].zeitfeld.value = zeit
        }
        ② setTimeout("Zeitauslese()",1000)
    </script>
</head>
<body>

```

```

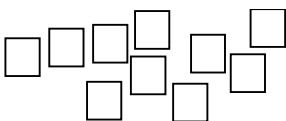
<h2>Das Auslesen der aktuellen Uhrzeit</h2>
<form>
  <input type="text" name="zeitfeld" value="" size="8" readonly>
</form>
<script>
  Zeitauslese();
</script>
</body>
</html>

```

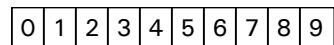
- ① In der Funktion `Zeitauslese` wird die aktuelle Uhrzeit bestimmt.
- ② Um die sekundengenaue Uhrzeit zu ermitteln, wird in der Funktion eine Verzögerung gesetzt, die nach einer Sekunde wieder dieselbe Funktion ausführt. Somit wird eine Endlosschleife zum Auslesen der Zeit geschaffen.
- ③ Dies ist das Eingabefeld, in dem das Ergebnis ausgegeben wird.
- ④ Um das Auslesen der aktuellen Uhrzeit zu starten, wird im Dokumentenkörper der Start der Endlosschleife gestartet, indem die Funktion beim Laden der Webseite das erste Mal aufgerufen wird.

Objekt **Array** für Variablenlisten

Ein Array (*Feld*) ist ein Speicherbereich, in dem Sie mehrere Variablen speichern können. Der Zugriff auf die Variablen erfolgt über einen Bezeichner und einen Index. Da JavaScript nicht typisiert ist, können in einem Array auch Werte verschiedenster Typs gespeichert werden. Um beispielsweise die Daten von 20 Personen zu speichern, benötigen Sie normalerweise 20 Variablen. Mit dem `Array`-Objekt können Sie diese in einem einzigen Array speichern.



Einfache Variablen



Aufbau eines Arrays

Objekterzeugung	<ol style="list-style-type: none"> ① <code>var Feldname = new Array();</code> ② <code>var Feldname = new Array(Anzahl der Elemente);</code> ③ <code>var Feldname = new Array(Element1, Element2, Element3, ...);</code>
------------------------	--

Sie können ein Array-Objekt auf drei verschiedene Arten erzeugen:

- ① Sie erzeugen ein leeres Feld mit einer unbestimmten Größe.
- ② Sie teilen beim Erzeugen des Objekts mit, wie viele Elemente es enthalten wird.

```
name = new Array(20); // das Feld wird 20 Elemente aufnehmen.
```

- ③ Sie können dem Array-Objekt bereits die gewünschten Elemente zuweisen.

```
// vier Namen werden zugewiesen
name = new Array("Müller", "Schmidt", "Schulze", "Lehmann");
```

Ein Objekt vom Typ `Array()` besitzt eine Eigenschaft und neun Methoden, um auf Felder zugreifen zu können.

Eigenschaften	<code>length</code>
Methoden	<code>concat()</code> , <code>join()</code> , <code>pop()</code> , <code>push()</code> , <code>reverse()</code> , <code>shift()</code> , <code>slice()</code> , <code>splice()</code> , <code>sort()</code>

Die Anzahl der in einem Feld befindlichen Elemente erhalten Sie über die Eigenschaft `length`. Der Zugriff auf ein einzelnes Element des Feldes erfolgt über die Angabe einer Indexnummer, die in eckigen Klammern angegeben wird. So erhalten Sie mit `name[0]` das erste Element des Feldes `name`, mit `name[1]` das zweite Element, das letzte Element erhalten Sie mit `name[name.length-1]`.

Mit den Methoden können Sie auf die einzelnen Elemente zugreifen und den Inhalt eines Feldes verändern.

Methode	Erläuterung
<code>concat(Array1, Array2, ...)</code>	Mit <code>concat()</code> können Sie dem ersten Feld die Inhalte weiterer Felder hinzufügen. Die neuen Inhalte werden hinten angehängt.
<code>join(Zeichen)</code>	Die einzelnen Elemente des Feldes werden durch das angegebene Zeichen miteinander verbunden und in einer Variablen abgelegt. Wird kein Verknüpfungszeichen angegeben, wird ein Komma verwendet. <code>a = new Array("Wind", "Regen", "Feuer")</code> <code>Var1 = a.join() // Var1 = "Wind,Regen,Feuer"</code> <code>Var2 = a.join(" + ") // Var2 = "Wind + Regen + Feuer"</code>
<code>pop()</code>	Liest das letzte Element im Feld aus und löscht es danach.
<code>push(Wert1, Wert2, ...)</code>	An das Ende des Feldes werden weitere Elemente angefügt. Zurückgegeben wird die neue Länge des Feldes.
<code>reverse()</code>	Kehrt die Reihenfolge der Elemente innerhalb des Feldes um. Das letzte Element wird zum ersten Element usw.
<code>shift()</code>	Das erste Element wird ausgelesen und gelöscht.
<code>slice(start, ende)</code>	Diese Methode gibt einen Teil des Feldes zurück, der mit dem Index <code>start</code> beginnt und mit dem Index <code>ende</code> endet. Das Ergebnis wird in einem neuen Array-Objekt zurückgegeben.
<code>splice(start, ende, Wert1, Wert2, ...)</code>	Die Elemente vom Index <code>start</code> bis <code>ende</code> werden durch die neuen Werte <code>Wert1, Wert2, ...</code> ersetzt. Als Rückgabewert erhalten Sie die Elemente von <code>start</code> bis <code>ende</code> .
<code>sort()</code>	Der Inhalt des Feldes wird alphabetisch sortiert. Enthalten die Elemente Zahlen, werden diese in Zeichenketten umgewandelt, sodass beispielsweise die Zahl 19 vor der Zahl 9 einsortiert wird. <code>name(1, 9, 19, 'Maria', 'Eva')</code> wird durch <code>name.sort();</code> sortiert zu <code>name(1, 19, 9, 'Eva', 'Maria')</code> .

Beispiel: Wochentags- und Monatsnamen (`kap07\date_array.html`)

In Zusammenhang mit dem Date-Objekt können Sie statt der bisherigen numerischen Tag- und Monatsausgabe in Verbindung mit einem Array-Objekt die entsprechenden Wochentags- und Monatsnamen zuweisen und ausgeben lassen.

```
<html>
<head>
  <meta charset="utf-8">
  <title>Datumsausgabe</title>
</head>
<body>
  <script>
    var Jetzt = new Date();
```

```

② var Tag    = Jetzt.getDate();
var WTag   = Jetzt.getDay();
var Monat = Jetzt.getMonth();
var Jahr   = Jetzt.getFullYear();

③ Wochentagname = new Array("Sonntag", "Montag", "Dienstag", "Mittwoch",
                            "Donnerstag", "Freitag", "Samstag");

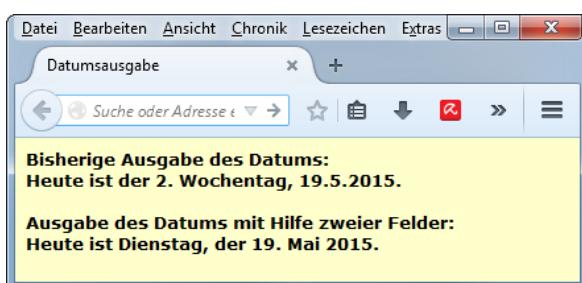
④ Monatsname = new Array("Januar", "Februar", "März", "April", "Mai", "Juni",
                         "Juli", "August", "September", "Oktober", "November", "Dezember");

⑤ document.write('Bisherige Ausgabe des Datums:<br>Heute ist der ' + WTag
⑥     + ' . Wochentag, ' + Tag + '.' + (Monat+1) + '.' + Jahr + '.<br><br>');
document.write('Ausgabe des Datums mithilfe zweier Felder:<br>Heute ist '
               + Wochentagname[WTag] + ', der ' + Tag + ' ' + Monatsname[Monat]
               + ' ' + Jahr + '.');

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

```

- ① Die Variable `Jetzt` wird als Objekt vom Typ `Date` deklariert.
- ② Den verschiedenen Variablen werden die Ergebnisse des aktuellen Datums übergeben.
- ③ Das Feld `Wochentagname` wird mit den Namen der Wochentage gefüllt.
- ④ Genauso verhält es sich mit dem Feld `Monatsname`. Es erhält als Elemente die Namen der zwölf Monate.
- ⑤ Das Datum mit den Rückgaben des `Date`-Objekts ohne die Zuweisung der Felder mit den Tages- und Monatsnamen wird angezeigt.
- ⑥ Die Ausgabe des Datums erfolgt in der Form: *Heute ist Dienstag, der 19. Mai 2015*. Die Variable `WTag` enthält in numerischer Form den aktuellen Tag (z. B. Freitag = 5). Mit dieser Nummer können Sie direkt das entsprechende Element des Feldes `Wochentagname` auslesen. Der Zugriff auf die Elemente beginnt bei 0 (null), sodass Sie mit der Indexnummer 5 das sechste Element erhalten. Das Auslesen des Monatsnamens aus dem Feld `Monatsname` verhält sich genauso. Der Zugriff erfolgt über den Wert der Variablen `Monat`.



Verbesserte Datumsangabe in Firefox

Der Verweis auf das Stylesheet, das für die Formatierung in der Abbildung verantwortlich ist, wurde im abgedruckten Code weggelassen.



7.9 Übung

Übung 1: Uhr

Übungsdatei: --

Ergebnisdatei: uebung1.html

1. Lesen Sie mit dem Date-Objekt die aktuelle Uhrzeit aus, und zeigen Sie diese kontinuierlich in Ihrem Dokument an. Verwenden Sie zur Aktualisierung eine Verzögerung von einer Sekunde.



JavaScript-Uhr

8 Zugriff auf HTML-Elemente

In diesem Kapitel erfahren Sie

- ✓ wie Sie auf die Unterobjekte des document-Objekts zugreifen
- ✓ wie Sie Grafiken mit JavaScript austauschen
- ✓ wie Sie auf Formulareingaben reagieren
- ✓ wie Sie ein Formular auf korrekte Eingaben kontrollieren

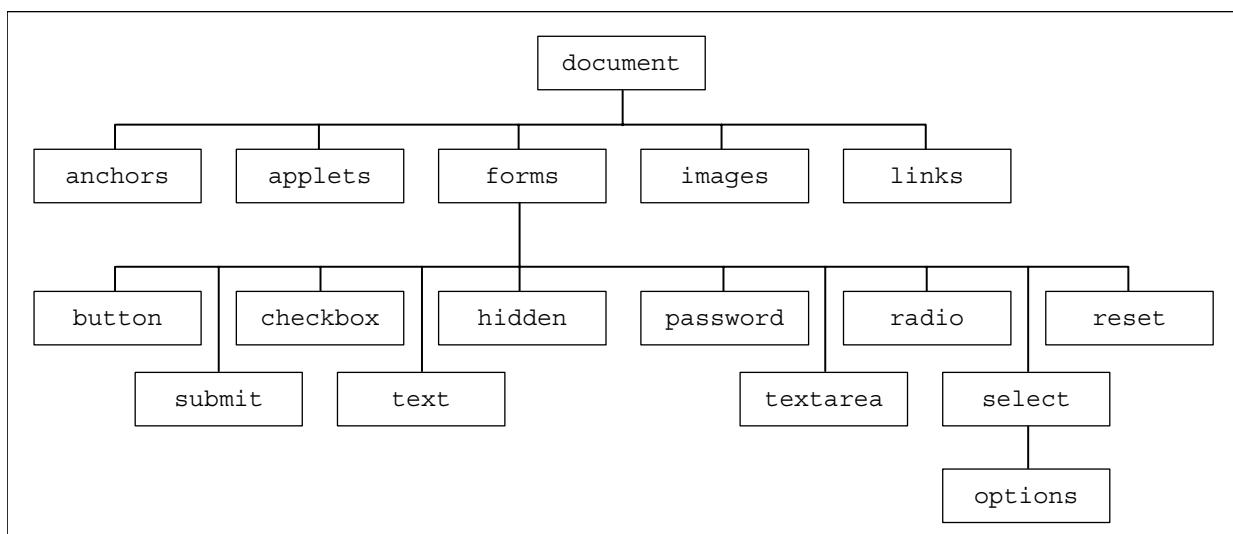
Voraussetzungen

- ✓ Formulare in HTML definieren
- ✓ Operatoren, Funktionen und Programmsteuerung
- ✓ Arbeiten mit Objekten

8.1 Grundlagen zum `document`-Objekt

Mit JavaScript können Sie auf HTML-Elemente zugreifen.

Ein `document`-Objekt ist ein Element des `window`-Objekts. Da es den Zugriff auf die Elemente einer Webseite ermöglicht, besitzt es weitere Unterobjekte. So können Sie über das Unterobjekt `forms` auf dessen Elemente und auf Formulareigenschaften zugreifen, sobald sich ein Formular `<form>...</form>` auf einer Webseite befindet.



Hierarchie des `document`-Objekts

Der Aufbau der Webseite bestimmt dabei das Vorhandensein von Instanzen eines Unterobjekts. Enthält beispielsweise ein HTML-Dokument kein Formular, besitzt das `document`-Objekt auch keine Objektreferenzen auf Formulare (`forms`).

8.2 Unterobjekt anchors

Mit diesem Objekt haben Sie Zugriff auf die Verweisanker einer Webseite. Über die folgende HTML-Anweisung wird z. B. ein Verweisanker definiert:

```
<a name="absatz2">&Uuml;berschrift zum 2. Absatz</a>
```

Folgende Eigenschaften sind für den Zugriff auf die Verweise implementiert.

Eigenschaften	length, name, text
---------------	--------------------

Mit der Abfrage `document.anchors.length` erhalten Sie die Anzahl der Verweisanker, die sich innerhalb einer Webseite befinden. Mit der Angabe eines Indexes können Sie den Namen des Ankers sowie den Verweistext auslesen: `document.anchors[0].name` oder `document.anchors[1].text`.

Die Eigenschaft `text` wird nicht vom Internet Explorer unterstützt. Es gibt für den Internet Explorer jedoch für diese Zwecke alternativ die Eigenschaft `innerText`.



8.3 Unterobjekt links

Um per JavaScript Zugriff auf die Verweise in einem HTML-Dokument zu haben, gibt es das Unterobjekt `links`. Es ist dem Unterobjekt `anchors` ähnlich und beinhaltet die gleichen Eigenschaften.

Eigenschaften	length, name, target, text
---------------	----------------------------

Die Eigenschaft `length` liefert Ihnen die Anzahl der im Dokument befindlichen Verweise und erlaubt Ihnen den Zugriff auf jeden einzelnen dieser Verweise. Der Zugriff auf die einzelnen Verweise erfolgt über sogenannte Indexnummern. Mit der Angabe des Indexes können Sie den Namen des Verweises sowie den Verweistext auslesen: `document.links[0].name` oder `document.links[1].text`. Das Zielfenster, in dem der Verweis geöffnet werden soll, erhalten Sie über die Anfrage der Eigenschaft `target`.

Die Eigenschaft `text` wird nicht vom Internet Explorer unterstützt.



Beispiel: Unterobjekt `links` (`kap08\objekt_links.html`)

```
<script>
  for(i = 0; i < document.links.length; i++) {
    alert((i + 1) + ". Link:" + "\n" +
          "Name: " + document.links[i].name + "\n" +
          "Target: " + document.links[i].target + "\n" +
          "Ziel: " + document.links[i] + "\n" +
          "Text: " + document.links[i].text + "\n");
  }
</script>
```

8.4 Unterobjekt applets

Dieses Objekt erlaubt den Zugriff auf Java-Programme, die speziell für den Gebrauch im Browser erstellt worden sind, sogenannte Java-Applets. Das Objekt `applets` besitzt eine Eigenschaft, um auf Java-Applets zugreifen zu können.

Eigenschaften	<code>length</code>
---------------	---------------------

Die Abfrage `document.applets.length` gibt die Anzahl der Java-Applets innerhalb eines HTML-Dokuments zurück.

8.5 Unterobjekt forms

Ein Formular wird auf einer Webseite verwendet, um Informationen zu sammeln und auszuwerten. Wie die Daten eingegeben werden, ist abhängig von den Eingabeelementen im Dokument. In Zusammenhang mit den Formularen in einem HTML-Dokument besitzt das `forms`-Objekt eine Reihe von Unterobjekten, die den Elementen des `<form>`-Tags in HTML entsprechen.

Formular

Ein Formular wird zur Eingabe von Daten verwendet, die nach dem Absenden verarbeitet werden sollen. Die Auswertung kann dabei auf unterschiedliche Weise erfolgen. Die einfachste Art ist es, die Daten per E-Mail an eine bestimmte Person zu senden. Oder sie werden an ein Programm, meist ein in den Programmiersprachen PHP oder Perl geschriebenes Skript, auf den Server geschickt und ausgewertet. In diesem Fall müssen die Adresse des Programms (`action`), der zu übertragende Datentyp (`encoding`) und das zu verwendende Übertragungsverfahren (`method`) bekannt sein.

Den Aufbau des Formulars finden Sie in den Eigenschaften und Methoden des `forms`-Objekts wieder.

Eigenschaften	<code>action, encoding, length, method, name, target</code>
Methoden	<code>reset(), submit()</code>
Unterobjekte	<code>elements</code>
Aufruf	<code>document.forms[Index].Eigenschaft</code> <code>document.forms[Index].Methode()</code> <code>document.FormularName.Eigenschaft</code> <code>document.FormularName.Methode()</code>

Den Namen des Formulars erhalten Sie über die Abfrage der Eigenschaft `name`. Das Zielfenster, in dem beispielsweise die Rückmeldung des Serverprogramms erscheinen soll, erhalten Sie mit `target`, das dem HTML-Attribut `<form target="">` entspricht. Um die Anzahl der im Formular befindlichen Elemente zu ermitteln, fragen Sie die Eigenschaft `length` ab. Mit den beiden Methoden können Sie die Eingaben des Formulars auf den ursprünglichen Zustand zurücksetzen (`reset()`) oder das Formular absenden (`submit()`).

Beispiel: Formulardaten auslesen ([kap08/formular_auslesen.html](#))

Es steht beispielsweise der folgende HTML-Quelltext als erstes Formular in einem HTML-Dokument.

```
<form name="Formular" action="test.php" method="post" target="_top">
</form>
```

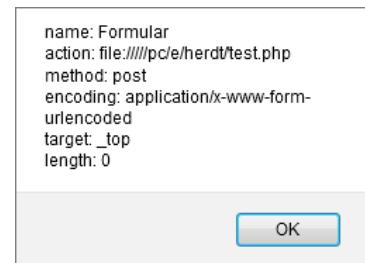
Der JavaScript-Code zum Auslesen dieser Formulardaten lautet:

```
<script>
  var text = "";
  text ='name: ' + document.forms[0].name + '\n'
    + 'action: ' + document.forms[0].action + '\n'
    + 'method: ' + document.forms[0].method + '\n'
    + 'encoding: ' + document.forms[0].encoding + '\n'
    + 'target: ' + document.forms[0].target + '\n'
    + 'length: ' + document.forms[0].length;
  alert(text);
</script>
```

Ausgabe der Formulardaten

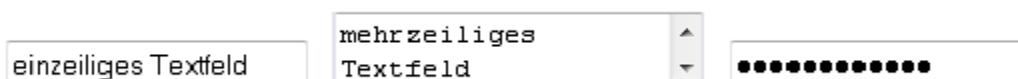
Sie können nicht nur die Eigenschaften des gesamten Formulars ermitteln. Die Funktionen eines Formulars verbergen sich in den einzelnen Elementen, die im Array `document.FormName.elements[]` .Eigenschaft des Formulars gespeichert sind. Mit der Angabe eines Indexes können Sie ein bestimmtes Element des Formulars ansprechen, z. B. das vierte Element über `document.FormName.elements[3]` .Eigenschaft. Wenn Sie den Namen des Elements kennen, können Sie es direkt ansprechen und dessen Eigenschaften abfragen bzw. ändern: z. B.

`document.FormName.ElementName.Eigenschaft = Wert.` Ist das Attribut `id` bekannt, kann eine Abfrage über `document.getElementById(id)` .Eigenschaft erfolgen.



8.6 Eingabefelder

In HTML-Dokumenten kann in diesen Feldern Text eingegeben werden. Zur Verfügung stehen hierbei drei verschiedene Arten von Eingabefeldern: `<input type="text">` für einzeilige Felder, `<textarea>` für mehrzeilige Felder und `<input type="password">` für maskierte Passworteingaben.



Unterschiedliche Arten von Eingabefeldern

Um die einzelnen Eingabefelder ansprechen zu können, existieren in JavaScript die jeweils entsprechenden Objekte `text`, `textarea` und `password`. Alle drei Objekte besitzen die nachfolgenden Eigenschaften und Methoden:

Eigenschaften	<code>defaultValue, form, name, type, value</code>
Methoden	<code>blur(), focus(), select()</code>

Die Eigenschaften erlauben das Auslesen eines Feldnamens (`name`) und des eingegebenen Wertes (`value`). Wird im HTML-Code für das Eingabefeld bereits ein standardmäßiger Wert angegeben, so wird dieser über die Eigenschaft `defaultValue` zurückgegeben. `type` gibt den Typ des Objekts zurück (`text`, `textarea`, `password`). Mit der Eigenschaft `form` ermitteln Sie den Namen des Formulars, in dem sich das Element befindet.

Die Methode `focus` setzt den Cursor in das entsprechende Eingabefeld, mit `blur` hingegen wird das Feld verlassen. Mit `select` können Sie den Inhalt des Feldes selektieren.

Versteckte Felder

Innerhalb eines Formulars können Sie mit der Angabe `<input type="hidden" . . .>` zusätzliche Informationen verschicken. Diese Felder werden vom Browser nicht angezeigt. Mit JavaScript können Sie folgende Eigenschaften des `hidden`-Elements abfragen:

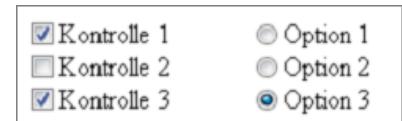
Eigenschaften	<code>form, name, type, value</code>
---------------	--------------------------------------

Mit diesen Eigenschaften haben Sie Zugriff auf den Namen des Formulars (`form`), den Namen des Objekts (`name`) und den Wert des versteckten Feldes (`value`). Die Abfrage, ob es sich um ein verstecktes Feld handelt, erfolgt über die Eigenschaft `type`.

Da dieses Element derzeit keine Methoden besitzt und deshalb nicht auf Ereignisabfragen reagieren kann, ist es für die weitere Verwendung in JavaScript nicht von Interesse.

8.7 Kontroll- und Optionsfelder

Eine Auswahl können Sie durch Kontroll- (engl. *checkbox*) oder Optionsfelder (engl. *radio buttons*) ermöglichen. Beide Elemente scheinen auf den ersten Blick die gleiche Funktion zu haben. Der Unterschied besteht darin, dass mehrere Kontrollfelder markiert werden können, während in den Optionsfeldern immer nur eine Markierung innerhalb einer Gruppe möglich ist.



Kontroll- und Optionsfelder

Den Zugriff auf diese Elemente erhalten Sie über die `checkbox`- und `radio`-Elemente, die folgende Eigenschaften besitzen:

Eigenschaften	<code>checked, defaultChecked, form, name, type, value</code> nur Optionsfeld: <code>length</code>
Methoden	<code>blur(), click(), focus()</code>

Mit der Eigenschaft `checked` erhalten Sie den Wahrheitswert, ob ein Kontrollfeld oder Optionsfeld markiert ist (`true` = ja, `false` = nein). In HTML können Sie mit dem Attribut `checked` festlegen, welche Option beim Laden der Webseite bereits gewählt sein soll. Die Eigenschaft `defaultChecked` liefert `true`, wenn das Element beim Laden bereits aktiviert ist, sonst `false`. Mit `form` erhalten Sie den Namen des Formulars. Den Namen des Kontroll- bzw. Optionsfeldes erhalten Sie mit `name`, das dem gleichnamigen HTML-Attribut entspricht. Den Wert des Feldes können Sie mit `value` auslesen (entspricht dem HTML-Attribut `value`). `type` gibt den Typ des Objekts zurück (`checkbox`, `radio`).

Das Optionsfeld besitzt noch eine zusätzliche Eigenschaft. Mit `length` können Sie die Anzahl der Optionsfelder innerhalb einer Optionsgruppe ermitteln.

Die Funktionalität der Kontroll- und Optionsfelder können Sie mit der Methode `click()` erweitern, um z. B. bei einer Auswahl entsprechend reagieren zu können. Mit `focus()` selektieren Sie ein Feld, mit `blur()` verlassen Sie es wieder.

8.8 Auswahllisten

Die Auswahllisten ermöglichen eine Selektion von einer oder mehreren (`multiple`) Optionen. Diese Listen werden in HTML mit dem HTML-Tag `<select>` erzeugt und in JavaScript durch das gleichnamige Objekt `select` repräsentiert.



Eigenschaften und Methoden des `select`-Objekts

Eigenschaften	<code>form, length, name, selectedIndex, type, value</code>
Methoden	<code>focus(), blur()</code>
Unterobjekt	<code>options []</code>

Die Anzahl der in der Auswahlliste befindlichen Optionen erhalten Sie über die Abfrage der Eigenschaft `length`. Ob eine Mehrfachauswahl in der Auswahlliste möglich ist, erfahren Sie über die Eigenschaft `type`. Die Rückgabe lautet entweder "select-one" für die einfache Auswahl oder "select-multiple" für die Mehrfachauswahl. `selectedIndex` gibt die Nummer der ausgewählten Option zurück, bei einer Mehrfachauswahl wird der erste ausgewählte Eintrag zurückgegeben.

Das `option`-Array

Die Einträge der Auswahlliste sind über das Feld `options []` erreichbar. Die Indexnummer, die den jeweiligen Eintrag der Auswahlliste ansprechen soll, wird in den eckigen Klammern angegeben. Die Elemente besitzen folgende Eigenschaften:

Eigenschaften	<code>defaultSelected, index, length, selected, text, value</code>
---------------	--

Den Wahrheitswert, ob eine Auswahl voreingestellt ist, erhalten Sie über die Eigenschaft `defaultSelected`. Die Nummer des Eintrags wird über `index` zurückgegeben. Ist eine Auswahl vom Benutzer vorgenommen worden, erhalten Sie diese über `selected`. Die Anzahl der möglichen Auswahlelemente erhalten Sie mit der Eigenschaft `length`. Den Text, der als Auswahl angezeigt wird, können Sie mithilfe von `text` ermitteln. Den Wert, der bei einer aktiven Option übermittelt wird, erfahren Sie über die Eigenschaft `value`.

Beispiel: Eigenschaften einer Auswahlliste anzeigen (`kap08\auswahlliste.html`)

Die Funktion `durchlauf()` zeigt alle Eigenschaften eines `option`-Elements als Textausgabe in einem mehrzeiligen Textfeld an. Dazu wird über eine `for`-Schleife eine Zeichenkette erzeugt, die die Eigenschaften enthält. Die Zeichenkette wird der Eigenschaft `value` des `textarea`-Elements zugewiesen. Die Zeilenumbrüche werden durch die Escapesequenzen `\n` erzeugt.

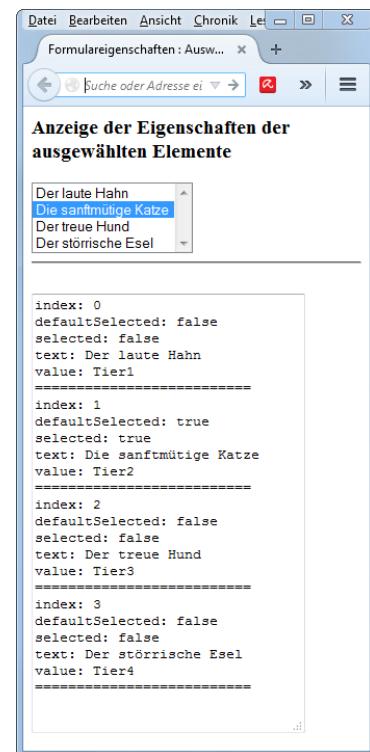
```
<script>
    function durchlauf() {
        var text = "";
        for (i = 0; i < document.formular.auswahl.length; i++) {
            text = text +
                'index: ' + document.formular.auswahl.options[i].index + '\n'
                + 'defaultSelected: ' +
                    document.formular.auswahl.options[i].defaultSelected + '\n'
                + 'selected: ' + document.formular.auswahl.options[i].selected + '\n'
                + 'text: ' + document.formular.auswahl.options[i].text + '\n'
                + 'value: ' + document.formular.auswahl.options[i].value + '\n' +
                '=====';
        }
        document.formular.ausgabe.value = text;
    }
</script>
```

JavaScript-Quellcode zur Ausgabe der Eigenschaften der einzelnen Listeneinträge

Die Auswahlliste `auswahl` befindet sich in einem HTML-Formular mit dem Namen `formular`. Die Funktion `durchlauf` wird nach dem Laden der Webseite aufgerufen (`onLoad="durchlauf () "`). Nach jeder geänderten Auswahl wird die Anzeige aktualisiert.

```
<body onload="durchlauf () ;">
<form action="" name="formular">
  <select name="auswahl" size="4" multiple
    onChange="durchlauf () ;>
    <option value="Tier1">Der laute Hahn</option>
    <option value="Tier2" selected>Die sanftmütige
      Katze</option>
    <option value="Tier3">Der treue Hund</option>
    <option value="Tier4">Der störrische
      Esel</option>
  </select>
  <hr>
  <textarea cols="30" rows="25" name="ausgabe">
  </textarea>
</form>
</body>
```

HTML-Quelltext der Auswahlliste



Ausgabe der Eigenschaften
(kap08\auswahlliste.html)

8.9 Schaltflächen

Eines der wichtigsten Elemente eines Formulars sind die Schaltflächen (engl. *buttons*). Sie dienen zum Start der Verarbeitung der eingegebenen Daten.

Eigenschaften	form, name, type, value
Methoden	click()

Die Eigenschaften erlauben das Auslesen des Formularnamens (`form`), des Schaltflächennamens (`name`), des Typs (`type`) sowie der Beschriftung der Schaltfläche (`value`).

Schaltflächen können mit der Ereignisabfrage `onClick` auf das Anklicken mit der Maus reagieren. Ansonsten bleiben diese Schaltflächen wirkungslos. Eine Ausnahme bilden die zwei Schaltflächen `Submit` und `Reset` zum Versenden und Zurücksetzen der eingegebenen Daten. Diese beiden standardmäßigen Schaltflächen dienen dazu, die eingegebenen Daten ohne Zuhilfenahme von JavaScript zu versenden oder zu löschen. Durch den Einbau einer Ereignisabfrage `onClick` wäre es beispielsweise möglich, die Daten vor dem Versenden auf Vollständigkeit zu prüfen bzw. beim Zurücksetzen eine Sicherheitsabfrage einzubauen.

8.10 Eingaben überprüfen

Die meistgenutzte Funktion des `forms`-Objekts ist die Überprüfung, ob die Felder eines Formulars ordnungsgemäß ausgefüllt wurden. Zu Beginn erstellen Sie ein Eingabefeld mit einer Schaltfläche `Submit` zum Absenden der Daten.

Beispiel: Formulareingabe prüfen (*kap08\form_tester.html*)

```
<body>
  <div align="center">
    <h3>Test einer Eingabe und des Wertes</h3>
    <p>Geben Sie eine beliebige Zahl ein:</p>
    <form action="" name="Eingabe" onSubmit="return testen() ;">
    <form action="#" name="Eingabe" onSubmit="return testen() ;">
      <input type="text" name="Feld">
      <input type="submit" name="x1" value="Testen">
    </form>
  </div>
</body>
```

Vor dem Absenden der Formulardaten wird das Ereignis `onSubmit` ausgelöst. Es ruft beim Betätigen der Schaltfläche `x1` die Funktion `testen` auf. Damit bei einer falschen Eingabe das Versenden der Formulardaten verhindert wird, gibt das Schlüsselwort `return` den Rückgabewert der Funktion zurück. Ist die Eingabe der Daten korrekt, liefert `return` den Wert `true` und das Versenden wird durchgeführt. Liefert die Funktion jedoch den Wert `false`, wird das Absenden der Daten unterbunden.

Der JavaScript-Code zur Kontrolle, ob Daten in das Feld eingegeben worden sind, sieht folgendermaßen aus:

```
<script>
  ① function testen() {
  ②   eingabe = document.Eingabe.Feld.value;
  ③   if (eingabe == "") {
  ④     alert("Sie müssen etwas eingeben!")
  ⑤     document.Eingabe.Feld.focus();
      return false;
    } else {
      alert("Vielen Dank!");
      return true;
    }
  }
</script>
```

- ① Die Funktion `testen`, die beim Betätigen der Schaltfläche zum Absenden der Daten aufgerufen werden soll, wird im Kopfbereich der Webseite definiert.
- ② Für die weitere Verarbeitung wird der Variablen `eingabe` der Wert des Eingabefeldes übergeben.
- ③ Die Kontrolle, ob etwas in das Feld eingegeben wurde, wird mit einer `if`-Abfrage realisiert. Dabei wird der Wert des Eingabefeldes ermittelt. Die Abfrage des Elements erfolgt über die Schablone `document.FormName.ElementName`. Eigenschaft und liest sich folgendermaßen: Wenn sich im Eingabefeld `Feld` des Formulars `Eingabe` des aktuellen Dokuments kein Wert befindet (`value`), soll eine Fehlermeldung ausgegeben werden.
- ④ Der Cursor wird mit der Methode `focus()` automatisch in das Eingabefeld gesetzt. Bei umfangreichen Formularen mit vielen Eingabefeldern erleichtern Sie somit dem Anwender die Suche nach dem fehlerhaften Eingabefeld. Mit `return false` wird die Funktion verlassen und der Wahrheitswert `false` zurückgegeben. Das Versenden der Daten wird verhindert.
- ⑤ Wurde etwas in das Feld eingegeben, wird in den `else`-Zweig der Funktion verzweigt. Er gibt eine Bestätigung an den Anwender aus und liefert an die Ereignisabfrage den Wert `true` zurück. Das Versenden der Formulardaten wird durchgeführt.

Zahleneingabe testen



Mithilfe des Objekts `String` haben Sie die Möglichkeit, die Daten eines Eingabefeldes auf bestimmte Zeichen zu kontrollieren. Das nachfolgende Beispiel testet den Wert eines Eingabefeldes auf numerische Daten. Ist die Eingabe eine Zahl, wird der Wahrheitswert `true` als Resultat zurückgegeben. Ist die Eingabe falsch, wird `false` zurückgeliefert.

```
function NumTest(eingabe) {
    var daten = "0123456789";
    for (i=0; i < eingabe.length; i++) {
        if (daten.indexOf(eingabe.charAt(i))<0 ) {
            alert(eingabe + ' ist keine gültige Zahl.');
            return false;
        }
    }
    return true;
}
```

Über die Variable `daten` werden die möglichen Eingabewerte hinterlegt, in diesem Fall die Zahlen von 0 bis 9. In der `for`-Schleife wird jeder Teil des Feldwertes `eingabe` daraufhin kontrolliert, ob er in der Variablen `daten` vorkommt. Sollte eine Stelle des Eingabewertes nicht korrekt sein, wird eine entsprechende Ausgabe getätigkt und das Absenden über die Rückgabe von `return false` verhindert.

Abfrage eines Optionsfeldes

Der Test, ob ein Optionsfeld gewählt wurde, ist etwas umfangreicher.

```
<form action="" name="RadioForm" onSubmit="return test();">
    <input type="radio" value="1" name="Abfrage">Option 1<br>
    <input type="radio" value="2" name="Abfrage">Option 2<br>
    <input type="radio" value="3" name="Abfrage">Option 3
</form>
```

- Option 1
- Option 2
- Option 3

Optionsfelder

Das grundlegende Problem bei dieser Abfrage ist, dass zuerst die Anzahl der Optionsfelder ermittelt werden muss. Danach müssen Sie jedes Optionsfeld mit der Eigenschaft `checked` daraufhin überprüfen, ob es ausgewählt wurde. Wenn keine der vorgegebenen Optionen gewählt wurde, soll eine Fehlermeldung erscheinen.

Beispiel: Optionsfeld abfragen (*kap08\form_tester_options.html*)

```
① function optionstest() {
    ②     var j = false;
    ③     for(i = 0; i < document.RadioForm.Abfrage.length; i++)
    ④         if(document.RadioForm.Abfrage[i].checked) {
    ⑤             j = true;
    ⑥             break;
    ⑦         }
    ⑧         if(!j) {
        ⑨             alert("Sie müssen mindestens eine Option auswählen.")
            ⑩             return false;
    ⑪     }
    ⑫     return true;
}
```

- ① Die Variable `j` wird mit dem Wahrheitswert `false` initialisiert und soll später den Wert `true` erhalten, sobald ein Optionsfeld gewählt wurde.
- ② Dies ist die Schleife, um jedes einzelne Optionsfeld anzusteuern. Die Eigenschaft `length` gibt hierbei die Anzahl der Optionsfelder Abfrage des Formulars `RadioForm` zurück. Die Variable `i` ist die Zählvariable.
- ③ Mithilfe der Variablen `i`, die in diesem Fall die Indexnummer darstellt, kann jedes einzelne Optionsfeld angesprochen werden. Ist eine Option vom Anwender gewählt worden, liefert die Eigenschaft `checked` den Wert `true` zurück und es wird in die `if`-Schleife verzweigt.
- ④ Die Variable `j` erhält den Wert `true`. Sie enthält somit die Information, dass eine Option ausgewählt wurde.
- ⑤ Mit `break` wird die `for`-Schleife vorzeitig verlassen, da nach dem Auffinden einer Auswahl die anderen, noch nicht kontrollierten Optionen nicht mehr getestet werden müssen.
- ⑥ Mit der Variablen `j` wird abgefragt, ob ihr Wahrheitswert `false` ist, also keine Option gewählt wurde. Ist dies der Fall, wird eine Meldung ausgegeben und die Funktion `optionstest` mit `return false` verlassen. Ist der Wahrheitswert von `j` gleich `true`, dann wird die Funktion weiter abgearbeitet, um beispielsweise weitere Formularelemente zu kontrollieren.

Sicherheitsabfrage beim Zurücksetzen

Ein weiteres Beispiel zur Arbeit mit den Formularen ist die Sicherheitsabfrage beim Zurücksetzen der eingegebenen Daten. Normalerweise werden alle Eingaben sofort gelöscht, sobald der Anwender die Schaltfläche vom Typ `reset` betätigt. Dies kann jedoch manchmal ungewollt sein. Die nachfolgende Funktion fragt den Anwender, ob die Daten wirklich gelöscht werden sollen.

```
function warnung() {  
    return confirm("Wollen Sie wirklich alle Eingaben zurücksetzen?");  
}
```

Diese Funktion soll das Zurücksetzen des Formulars abfangen und ein Dialogfenster zum Bestätigen des Vorgangs einblenden. Bestätigt der Anwender das Löschen der Daten, liefert `return` den Wahrheitswert `true` zurück und das Zurücksetzen der Daten wird durchgeführt. Möchte er das Löschen abbrechen, wird `false` zurückgeliefert und die eingegebenen Daten bleiben erhalten.

Diese Abfragefunktion rufen Sie beim Zurücksetzen des Formulars mit der Ereignisabfrage `onReset` im `<form>`-Tag auf.

```
<form onReset="return warnung();">  
</form>
```

8.11 Unterobjekt `images`

Um die Eigenschaften von Grafiken auslesen und ändern zu können, wird das Objekt `images` benutzt. Der Browser erstellt dieses Objekt automatisch, sobald auf einer Webseite eine Grafik geladen wird.

Eigenschaften	<code>border, complete, height, hspace, length, lowsrc, name, src, vspace, width</code>
Methode	<code>handleEvent()</code>
Aufruf	<code>document.images[Index].Eigenschaft</code> <code>document.images[Index].Methode()</code> <code>document.BildName.Eigenschaft</code> <code>document.BildName.Methode()</code>

Wie jedes Objekt besitzt auch `images` die Eigenschaft `length`, um die Anzahl der im Dokument befindlichen Grafiken zu ermitteln. Der Zugriff auf die einzelnen Grafiken erfolgt über die entsprechende Indexnummer. Den Index geben Sie hinter dem Objektnamen `images` in eckigen Klammern an: `document.images[0]`. Sie können eine Grafik auch direkt mit dem Namen ansprechen, den Sie zuvor mit dem Attribut `name` zugewiesen haben. Beispielsweise sprechen Sie mit `document.BildName` die Grafik `` an. Wurde der Grafik eine ID zugewiesen (``), können Sie diese auch über `document.getElementById()` ansprechen.

Für Grafiken, die Sie nachträglich mit JavaScript anzeigen möchten, müssen Sie jedoch eigene Grafikobjekte erzeugen. Das ist beispielsweise dann notwendig, wenn Sie per JavaScript einzelne Grafiken durch andere Grafiken ersetzen lassen wollen.

Objekterzeugung	<code>Bild1 = new Image();</code> <code>Bild1.src = 'bild.jpg';</code> <code>document.images[0].src = Bild1.src;</code>
------------------------	---

Zum Erzeugen eines Objekts vergeben Sie einen Variablenamen, den Sie mit `new Image()` initialisieren. Beachten Sie, dass der erste Buchstabe von `Image` großgeschrieben werden muss. Dem erzeugten Objekt können Sie nun eine Eigenschaft zuweisen, beispielsweise mit `src` die entsprechende Grafikdatei.

Austauschen von Grafiken

Ein sehr häufig genutzter Effekt im Internet ist der „MouseOver-Effekt“". Er wird deshalb so genannt, weil sich eine Grafik ändert, sobald sich die Maus über einem Link oder einer Grafik befindet.



Anzeige verschiedener Grafiken beim Überfahren unterschiedlicher Verweise mit der Maus

Beispiel: Bilder bei Mouseover austauschen (*kap08\images_bildtausch.html*)

Zuerst erstellen Sie den HTML-Code mit einer unsichtbaren Grafik und den Hyperlinks. Die nicht sichtbare Grafik dient als Platzhalter für die dynamisch erscheinende Grafik. Beim Überfahren der Hyperlinks mit der Maus wird statt der Platzhalter-Grafik eine andere Grafik angezeigt.

```

<html>
<head>
    <meta charset="utf-8">
    <title>Grafik austauschen</title>
</head>
<body>
    <div align="center">
        <h3>Grafik austauschen</h3>
        ① <br>
        ② <a href="#" onMouseOver="document.bild.src='images/fruehling.gif';"
            onMouseOut="zurueck();">Fr&ampuumlhling</a> -
        ③ <a href="#" onMouseOver="document.bild.src='images/sommer.gif';"
            onMouseOut="zurueck();">Sommer</a> -
        ④ <a href="#" onMouseOver="document.bild.src='images/herbst.gif';"
            onMouseOut="zurueck();">Herbst</a> -
        <a href="#" onMouseOver="document.bild.src='images/winter.gif';"
            onMouseOut="zurueck();">Winter</a>
        <p>F&ampuumlhren Sie die Maus über die Verweise, <br>um die
            verschiedenen Grafiken anzuzeigen.</p>
    </div>
</body>
</html>
```

- ① Die Grafik *images/blank.gif* ist unsichtbar, d. h., sie besitzt eine Farbe, die dem Hintergrund der Webseite entspricht. Die Farbe kann auch transparent sein. Die Grafik erhält die Bezeichnung *bild*.
- ② Der Link Frühling wird erstellt. Durch die leere Ankerangabe # wird der Link zwar vom Browser ausgeführt, es wird aber nicht auf eine andere Seite verzweigt. Über das Ereignis *onMouseOver* legen Sie fest, dass die Quelle (src) des Objekts *bild* durch die Grafik *images/fruehling.gif* ersetzt werden soll.
- ③ Sobald der Hyperlink verlassen wird (*onMouseOut*), setzen Sie die Grafik über die Funktion zurück.
- ④ Auf dieselbe Weise legen Sie für die weiteren Hyperlinks die anzugegenden Bilder fest.

Die folgende Funktion fügen Sie in den Kopfbereich der Webseite ein. Die Funktion *zurueck* ändert die Quelle der Grafik mit der Bezeichnung *bild*. Der Browser zeigt daraufhin wieder die unsichtbare Grafik *blank.gif* an.

```

<script>
    function zurueck() {
        document.bild.src = "images/blank.gif"
    }
</script>
```

Beispiel: Diashow (*kap08\images_diashow.html*)

Mit diesem Beispiel realisieren Sie eine Diashow, wobei die Reihenfolge der Anzeige mithilfe einer Navigation steuerbar ist.

```
<html>
<head>
    <meta charset="utf-8">
    <title>Bildwechsel</title>
</head>
<body>
    <div align="center">
        <h2>Dia-Show</h2>
        <form action="" name="legende">
            <input type="button" value="<<" onClick="wechsel(0);">
            <input type="button" value=" < " onClick="wechsel(1);">
            <input type="button" value=" > " onClick="wechsel(2);">
            <input type="button" value=">>" onClick="wechsel(3);"><br>
            <input type="Text" name="ausgabe" size="30" readonly>
        </form>
        <p>
    </div>
</body>
</html>
```

Die Steuerung der Grafikanzeige realisieren Sie über Formular-Schaltflächen. Sie können zu diesem Zweck auch Hyperlinks einsetzen. Beide unterstützen die Ereignisabfrage `onClick`, mit der der Grafikwechsel ausgelöst wird. Übergeben wird die Nummer der Schaltfläche, damit in der Funktion eine Unterscheidung für die anzugeigende Grafik vorgenommen werden kann.

Es folgt der JavaScript-Code zum Steuern der Anzeige der einzelnen Grafiken und Erläuterungen, den Sie in den Kopfbereich der Webseite einfügen.

	<pre><script> ① var bilder = new Array("blau.jpg", "land.jpg", "winter.jpg", "wueste.jpg"); ② var bildlegende = new Array("Das blaue Meer.", "Alle Vöglein sind schon da", "Sonnenuntergang im Winter", "Fata Morgana"); ③ var count = 0; ④ function wechsel(stelle) { if (stelle == 0) count = 0; if ((stelle == 1) && (count > 0)) count--; if ((stelle == 2) && (count < bilder.length - 1)) count++; if (stelle == 3) count = bilder.length - 1; ⑤ document.bild.src = "images/" + bilder[count]; ⑥ document.legende.ausgabe.value = bildlegende[count]; } </script></pre>
--	---

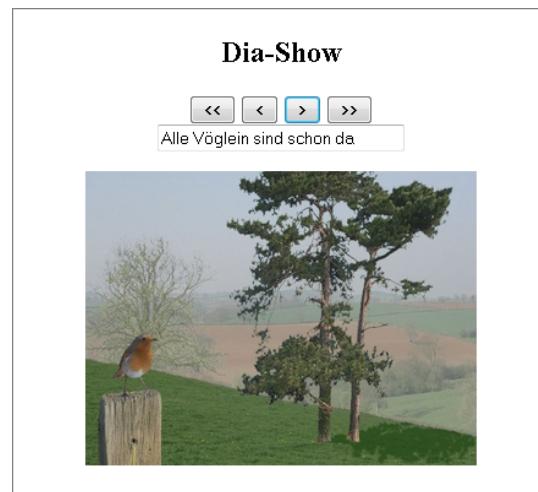
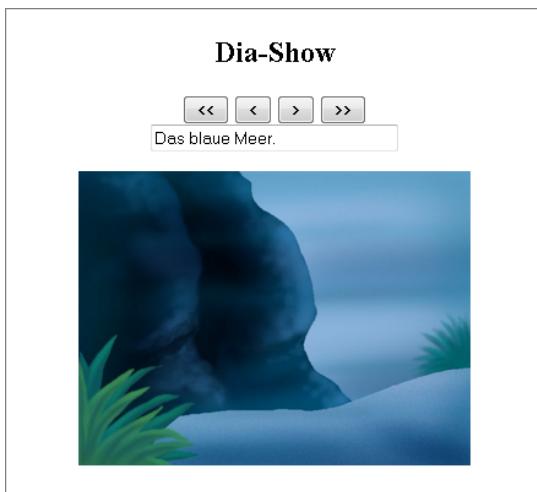
- ① Als Erstes erstellen Sie ein Feld, das die Dateinamen der anzugeigenden Grafiken enthält.
- ② In Anlehnung an das Feld `bilder` werden im Feld `bildlegende` die einzelnen Bildunterschriften angelegt, die beim Anzeigen des jeweiligen Bildes erscheinen sollen.

- ③ Die Variable `count` wird zum Zählen benötigt, um zu ermitteln, welches Bild aus dem Feld `bilder` gerade angezeigt wird.
- ④ Die Funktion `wechsel()` kontrolliert anhand des Übergabewertes `stelle`, auf welchen Wert der Zähler `count` gesetzt werden muss. Entweder wird der Zähler auf das erste Bild zurückgesetzt (`count = 0;`), um einen Wert erniedrigt (`count--;`), um einen Wert erhöht (`count++;`) oder auf das letzte Bild gesetzt (`count = bilder.length - 1;`).
- ⑤ Hier wird die anzuseigende Grafikdatei aus dem Feld `bilder` dem `images`-Objekt übergeben. Die Grafik, die angezeigt wird, hängt von der Zählvariablen `count` ab. Mit `document.bild.src` wird das bisherige Bild, das den Namen `bild` besitzt, überschrieben.
- ⑥ Im Eingabefeld `ausgabe` des Formulars `legende` wird die zum Bild gehörige Bildunterschrift ausgegeben. Die Bildunterschrift ist im Feld `bildlegende` gespeichert und wird ebenfalls mithilfe des Zählers `count` ausgelesen.

An das Ende der Webseite `images_diashow.html` fügen Sie den folgenden JavaScript-Code ein:

```
<script>
  wechsel(0);
</script>
```

Damit rufen Sie die Funktion `wechsel()` auf, zeigen die erste Grafik an und füllen nach dem Laden der Webseite das Eingabefeld mit der Erläuterung zum Bild.



Anzeige einer Grafik (`kap08\images_diashow.html`)

Wechsel zu einer anderen Grafik

8.12 Übungen

Übung 1: Länge der Texteingabe überprüfen

Übungsdatei: --

Ergebnisdatei: uebung1.html

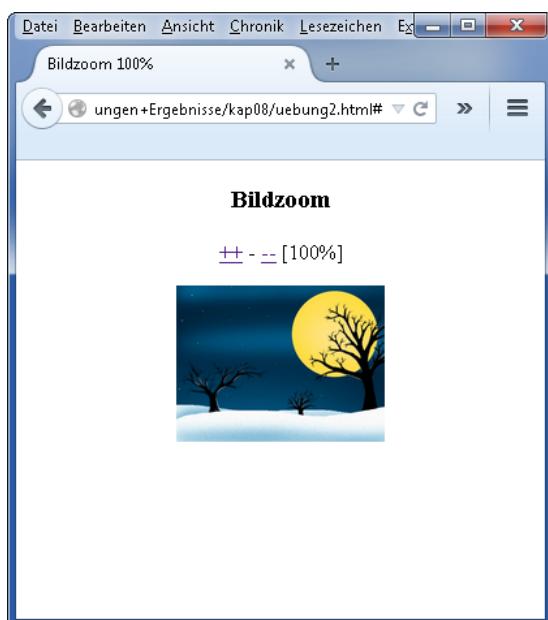
1. In einem Formularfeld soll bei der Texteingabe die Länge des bereits eingegebenen Textes ermittelt und angezeigt werden.
2. Geben Sie beim Verlassen des Eingabefeldes eine Warnung aus, wenn die Länge des eingegebenen Textes unterschritten bzw. überschritten wurde.

Übung 2: Größe einer Grafik ändern

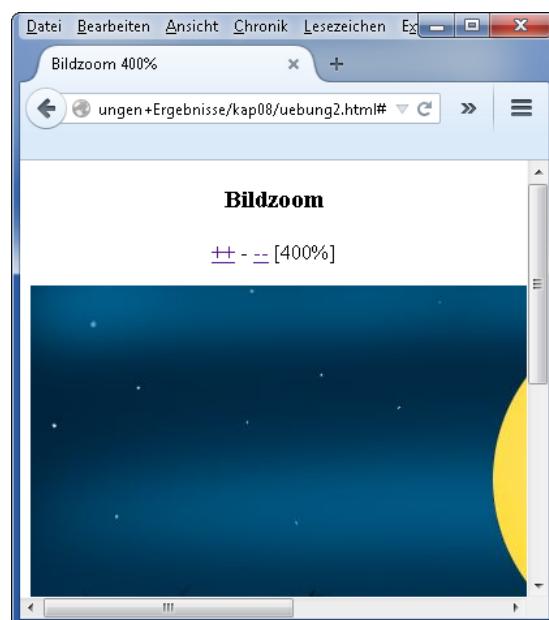
Übungsdatei: --

Ergebnisdatei: uebung2.html

3. Erstellen Sie eine Funktion, mit der Sie die Größe und Breite (`height` und `width`) eines Bildes verändern können. Dabei soll sich bei einem Klick auf einen Hyperlink die Größe jeweils verdoppeln bzw. halbieren. Eine Textanzeige soll den aktuellen Zoomfaktor darstellen.



Standardgröße der Grafik



2-fach vergrößertes Bild mit geänderter Titelleiste

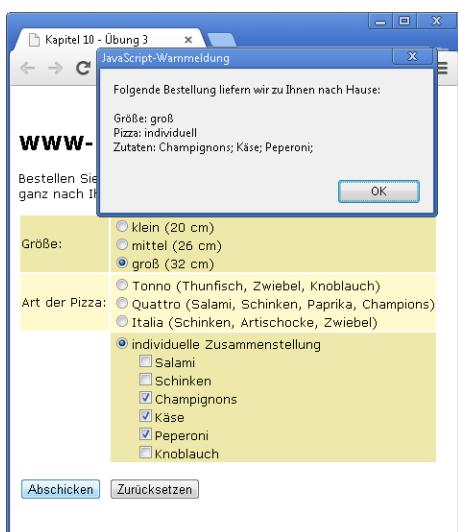
Übung 3: Formulareingaben auswerten

Übungsdatei: --

Ergebnisdatei: uebung3.html

1. Schreiben Sie ein Skript für die Bestellung einer Pizza.

- ✓ Der Kunde hat die Auswahl aus drei verschiedenen Pizzagrößen. Außerdem kann er entscheiden, ob er eine der drei Pizzas der Speisekarte wählt oder doch lieber eine individuelle Zusammenstellung wünscht.
- ✓ Wählt er die individuelle Pizza, muss er mindestens zwei Zutaten auswählen.
- ✓ Vor dem Verschicken der Daten soll kontrolliert werden, ob die Auswahl gültig ist.
- ✓ Der Kunde soll eine Meldung mit dem Ergebnis der Kontrolle erhalten.



Meldung zu gültiger Bestellung

9 Dynamic HTML

In diesem Kapitel erfahren Sie

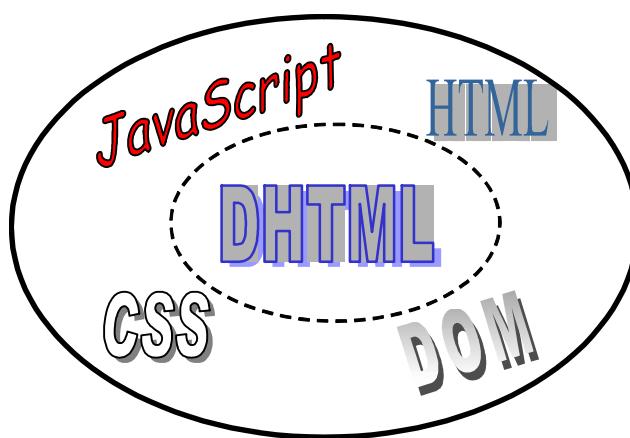
- ✓ wie Sie Ebenen anlegen
- ✓ wie Sie die Ebenen anzeigen und verbergen
- ✓ wie die Eigenschaften der Ebenen verändert werden
- ✓ wie Ebenen per JavaScript bewegt werden

Voraussetzungen

- ✓ Browser mit DOM Level 3 Standard
- ✓ Kenntnisse in CSS

9.1 Bestandteile des dynamischen HTML

Dynamic HTML (dynamisches HTML), kurz DHTML, ist kein festgelegter Standard einer neuen Skriptsprache, sondern eine Kombination aus HTML, CSS, JavaScript und dem DOM (Document Object Model).



Die Elemente des dynamischen HTML

Das Document Object Model versucht hierbei, alle relevanten Bestandteile einer angezeigten Webseite als Objekthierarchie abzubilden. Dabei fasst es die Möglichkeiten der objekt- und ereignisorientierten Programmierung zusammen. Das Modell legt beispielsweise fest, welche Komponenten eines HTML-Dokuments für JavaScript zugänglich sind. JavaScript regelt dabei, wie der Zugriff erfolgt, und das DOM gibt vor, auf was zugegriffen werden darf.

Die Spezifikation des Document Object Models (DOM Level 3 Standard) vom W3-Konsortium finden Sie unter <http://www.w3.org/DOM/>.

9.2 DOM-Standard ermitteln

Der Browser Internet Explorer (und der seit 2008 nicht mehr weiterentwickelte Netscape-Browser) hat in früheren Versionen die Elemente einer Webseite über ein jeweils eigenes Dokumentenobjektmodell angesprochen. Daher musste damals geprüft werden, welches Objektmodell verwendet werden kann. Dies ist heutzutage nicht mehr notwendig, da sich alle Browserhersteller an den W3C-Standard halten.

	Browserversion	Ansprechen des DOM über
①	Internet Explorer ab 4.x	<code>document.all</code>
②	Netscape Navigator 4.x	<code>document.layers</code>
③	Internet Explorer ab 5.0 Opera ab 6.0, Firefox ab 1.0, Safari und Chrome	<code>document.documentElement</code>

- ① Die Eigenschaft einer Ebene wurde im Internet Explorer der Version 4 über das `document.all`-Objekt angesprochen. Auch in der zur Zeit der Bucherstellung aktuellen Version 11 können Sie die Objekte des Dokumentenmodells noch über `all` ansprechen.
- ② Im Netscape Navigator 4.0 bis 4.7 wurde die Eigenschaft einer Ebene über das `document.layers`-Objekt und den Namen der Ebene angesprochen.
- ③ Die aktuellen Versionen der Browser nutzen zum Ansprechen der Elemente den Document Object Model Standard des W3-Konsortiums. Dies ist daher die einzige Methode, die heutzutage noch von Interesse ist.

Beispiel: Verwendung des aktuellen DOM prüfen (*kap09\browsertest.html*)

Statt wie bisher den Browser über das Auslesen seines Namens zu ermitteln, genügt es bei der Implementierung von dynamischem HTML, das verwendete Dokumentenobjektmodell abzufragen. Dieses Vorgehen wird auch als Browserweiche bezeichnet.

```

① ns4 = (document.layers) ? 1 : 0;
② ie4 = (document.all) ? 1 : 0;
③ w3c = (document.documentElement) ? 1 : 0;
④ if (w3c)
    alert("Sie benutzen einen Browser, der den aktuellen DOM-Standard unterstützt
          \n\nDies können sein:\nFirefox V1 oder höher\nMicrosoft Internet Explorer V5
          oder höher\nOpera V6 oder höher bzw. Apple Safari oder Chrome Google");
else if (ie4)
    alert("Sie benutzen den Microsoft Internet Explorer V4.x");
else if (ns4)
    alert("Sie benutzen den Netscape Navigator V4.x");
else
    alert("Leider kann ich Ihnen nicht sagen, welchen Browser Sie benutzen!");

```

- ① Die globale Variable `ns4` enthält den Wert 1, wenn der Netscape Navigator verwendet wird. Versteht der Browser die Anweisung `document.layers`, so handelt es sich um den Navigator in der Version 4.
- ② Genauso wird mit der Variablen `ie4` für den Internet Explorer verfahren. Hier wird getestet, ob der Browser den Befehl `document.all` versteht.
- ③ Als Letztes testen Sie, ob der Browser die Anweisung `document.documentElement` interpretieren kann. Dementsprechend wird die Variable `w3c` mit dem jeweiligen Wert belegt.
- ④ Über die Abfrage der einzelnen Variablen können Sie die Versionen und die Namen der verwendeten Browser bestimmen.

9.3 Container erstellen

<code><div>...</div></code>	Alle Browser der neuen Generation stellen Ihnen zum Erstellen eines Containers das Tag <code><div></code> zur Verfügung.
---	--

Um mehrere Elemente einer Webseite in allen Browsern ansprechen zu können, setzen Sie diese in einen Container, den Sie mit dem Tag `<div>...</div>` umschließen. Den in den `<div>`-Tags eingeschlossenen Elementen können die verschiedensten Stylesheet-Angaben zugewiesen werden. Die wichtigsten Eigenschaften sind jedoch die Positionierungseigenschaften und der Name der Ebene.

<pre><div id="ContainerName" style="position:...; left:...; top:...; visibility:...; z-index:...;"> Inhalt der Ebene </div></pre>

9.4 Ebenen anzeigen und verbergen

Eine der wichtigsten Eigenschaften von Ebenen ist es, dass sie angezeigt oder verborgen werden können. Die Cascading Style Sheets stellen Ihnen hierzu die Eigenschaft `visibility` zur Verfügung, die Sie mit JavaScript zur Laufzeit ändern können. Dazu selektieren Sie das HTML Element und weisen ihm mit der Eigenschaft `visibility` einen neuen Wert zu.

Beispiel: Ebenensichtbarkeit ändern ([kap09\visibility.html](#))

Beim Laden der Webseite wird bereits die Version des Browsers abgefragt und in den Variablen gespeichert. Die Funktionen `Sichtbar` und `Unsichtbar` sollen später über Hyperlinks aufgerufen werden und vorgegebene Ebenen anzeigen oder verbergen.

<pre><script> ① var w3c = (document.documentElement) ? 1 : 0; ② function Sichtbar(name) { if (w3c) document.getElementById(name).style.visibility = "visible"; else alert('Sie verwenden einen Browser, der den W3C-Standard zum Ansprechen von Elementen nicht unterstützt.'); } ③ function Unsichtbar(name) { if (w3c) document.getElementById(name).style.visibility = "hidden"; else alert('Sie verwenden einen Browser, der den W3C-Standard zum Ansprechen von Elementen nicht unterstützt.'); } </script></pre>
--

- ① Als Erstes folgt die Abfrage, ob der Browser das Modell nach dem W3C-Standard unterstützt. Die Variable `w3c` wird entsprechend mit dem Wert 1 oder 0 gefüllt.

- ② Die Funktion zum Anzeigen einer Ebene wird angelegt. Ihr soll später der Wert name zugewiesen werden, der den Namen oder die Indexnummer der zu ändernden Ebene enthält. Um die Ebene sichtbar zu machen, wird für die aktuellen Browser die Eigenschaft visibility auf den Wert visible gesetzt. Für ältere Browser soll eine entsprechende Meldung eingeblendet werden.
- ③ Die Funktion Unsichtbar() ist das Gegenteil der vorherigen Funktion. Der Wert hidden der CSS-Eigenschaft visibility lässt die Ebene verschwinden. Auch hier wird, wenn notwendig, auf den veralteten Browser hingewiesen.

Legen Sie zwei verschiedene Ebenen mit dem HTML-Tag <div> an. Um sie ansprechen zu können, weisen Sie ihnen jeweils einen eindeutigen Namen zu.

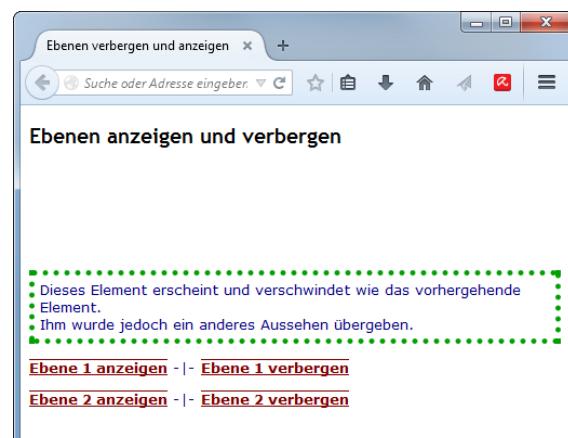
```

<body>
  <h2>Ebenen anzeigen und verbergen</h2>
  ① <div id="ebene1" style="visibility:hidden;">
    <p class="box1">Dieses Element wurde durch den Aufruf von
      "visibility=visible" sichtbar gemacht.<br>Wird die Sichtbarkeit auf
      "hidden" gesetzt, verschwindet das Element wieder.</p>
  </div>
  ② <div id="ebene2" style="visibility:hidden;">
    <p class="box2">Dieses Element erscheint und verschwindet wie das
      vorhergehende Element.<br>Ihm wurde jedoch ein anderes Aussehen
      übergeben.</p>
  </div>
  ③ <p><a href="javascript:Sichtbar('ebene1');">Ebene 1 anzeigen</a> - | -
    <a href="javascript:Unsichtbar('ebene1');">Ebene 1 verbergen</a></p>
  <p><a href="javascript:Sichtbar('ebene2');">Ebene 2 anzeigen</a> - | -
    <a href="javascript:Unsichtbar('ebene2');">Ebene 2 verbergen</a></p>
</body>
```

- ① Die erste Ebene wird mit dem Namen ebene1 angelegt. Beim Laden der Webseite wird diese nicht angezeigt, da in der Stylesheet-Formatierung die Sichtbarkeit mit visibility:hidden auf unsichtbar gesetzt wurde.
- ② Dies ist die zweite Ebene mit der Kennung ebene2, die ebenfalls nach dem Laden unsichtbar ist.
- ③ Die jeweilige Funktion Sichtbar() bzw. Unsichtbar() zum Setzen der Sichtbarkeit einer Ebene wird durch das Auslösen eines Verweises aufgerufen. Die Ebene, die angesprochen werden soll, wird als Wert an die Funktion übergeben (ebene1 oder ebene2). Damit wird innerhalb der Funktionen die Sichtbarkeit der gewünschten Ebene verändert.



In Firefox: Beide Ebenen sind sichtbar



In Firefox: Ebene 1 ist unsichtbar

9.5 Die Position von Ebenen zur Laufzeit ändern

Ebenen können Sie über die Stylesheet-Eigenschaften `top` (*oben*) und `left` (*links*) positionieren. So wie Sie die Sichtbarkeit von Ebenen per JavaScript ändern können, ist auch die Änderung der Position zur Laufzeit möglich.

Beispiel: Ebenenposition ändern (*kap09\positionieren.html*)

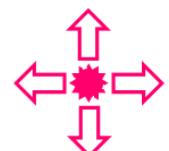
Als Beispiel wird eine Grafik nach jedem Klick auf ihre Pfeile um jeweils 25 Pixel in die entsprechende Richtung verschoben. Das Grundgerüst der Webseite sieht folgendermaßen aus.

```
<html>
<head>
    <meta charset="utf-8">
    <title>Ebenen positionieren</title>
</head>
① <body onLoad="init();">
    <h2>Ebenen positionieren</h2>
    <p>Klicken Sie auf einen der Pfeile, um die Ebene mit der Grafik an eine andere Stelle zu bewegen.</p>
② <div id="pfeile" style="position:absolute; left:200px; top:200px;
    visibility:visible;">
</div>
③ <map name="pfeilmap">
④     <area shape="circle" coords=" 25,65,25" href="javascript:scroll(25,0)">
        <area shape="circle" coords="109,65,25" href="javascript:scroll(25,1)">
        <area shape="circle" coords=" 67,25,25" href="javascript:scroll(25,2)">
        <area shape="circle" coords="67,107,25" href="javascript:scroll(25,3)">
</map>
</body>
</html>
```

- ① Während des Ladens der Webseite wird die Funktion `init()` aufgerufen, und die benötigten Variablen werden initialisiert.
- ② Eine Ebene wird mit der ID `pfeile` angelegt. Damit die Koordinaten `left` und `top` angewendet werden, müssen Sie zusätzlich die Positionierung festlegen. In diesem Fall werden die Koordinaten absolut zur linken oberen Browsserecke angegeben.
- ③ Innerhalb der Ebene befindet sich eine Grafik, die als Imagemap (Grafik mit Verweisen) benutzt werden soll.
- ④ Die Definition der Imagemap wird angelegt. Sie enthält vier Verweise, bei denen die Funktion `scroll()` mit verschiedenen Werten aufgerufen werden soll, sobald der Anwender einen Verweis in der Grafik auswählt.

Ebenen positionieren - Verschieben bei Klick

Klicken Sie auf einen der Pfeile, um die Ebene mit der Grafik an eine andere Stelle zu bewegen.



Die Grafik bewegt sich beim Klick auf einen Pfeil

Die Funktion zur Initialisierung der Werte und die Funktion zum Bewegen der Ebene legen Sie im Kopfbereich der Webseite fest.

```

① <script>
    var ebene;

② function init() {
    w3c = (document.documentElement) ? 1 : 0;
    if (w3c)
        ebene = document.getElementById('pfeile');
    else
        alert('Sie verwenden einen Browser, der den W3C-Standard zum Ansprechen
              von Elementen nicht unterstützt.');
}

④ function scroll(laufweite,richtung) {
    if (richtung==0)          // nach links
        ebene.style.left = (parseInt(ebene.style.left) - laufweite) + "px";
    else if (richtung==1)     // nach rechts
        ebene.style.left = (parseInt(ebene.style.left) + laufweite) + "px";
    else if (richtung==2)     // nach oben
        ebene.style.top = (parseInt(ebene.style.top) - laufweite) + "px";
    else if (richtung==3)     // nach unten
        ebene.style.top = (parseInt(ebene.style.top) + laufweite) + "px";
}
</script>

```

- ① Als Erstes wird die Variable `ebene` global definiert, um sie in beiden Funktionen ansprechen und verwenden zu können.
- ② In der Funktion `init`, die direkt nach dem Laden der Webseite aufgerufen wird, wird ermittelt, ob der Browser das aktuelle DOM umsetzen kann.
- ③ Der globalen Variablen `ebene` wird das Objekt des definierten Containers `pfeile` zugewiesen, um auf dessen Eigenschaften zugreifen zu können.
- ④ Die Funktion `scroll()` übernimmt die übergebenen Werte in die Variablen `laufweite` und `richtung`. Die Laufweite bestimmt dabei die Anzahl der Pixel, um die das Bild schrittweise bewegt werden soll. Die Richtung enthält die Information, ob die Grafik nach links, rechts, oben oder unten verschoben wird.
- ⑤ Es folgt eine Abfrage, ob die Grafik nach links bewegt werden soll. Zu diesem Zweck wird die Variable `richtung` daraufhin kontrolliert, ob sie den Wert 0 enthält.
- ⑥ Ist dies der Fall, wird die linke Ebenenkoordinate `ebene.style.left` neu berechnet. Hierbei wird der aktuelle Wert der Koordinate um den Wert der `laufweite` verringert. Da die Eigenschaft `ebene.style.left` eine Zeichenkette mit einer Maßeinheit zurückliefert, muss sie mit der Funktion `parseInt` in eine Ganzzahl umgewandelt werden. Das mathematische Ergebnis wird zum Schluss wieder in eine Zeichenkette umgewandelt, indem die Maßeinheit `px` hinten ange stellt wird.
- ⑦ Trifft die Bedingung ⑤ nicht zu, wird an dieser `else`-Anweisung verzweigt und kontrolliert, ob sich die Grafik nach rechts bewegen soll (`richtung==1`). In diesem Zweig werden die Koordinaten für die Bewegung nach rechts übergeben.
- ⑧ Diese Zeile enthält die Abfrage zur Bewegung der Grafik nach oben.
- ⑨ Hier wird kontrolliert, ob die Grafik nach unten bewegt werden soll.

Die erweiterte Form der bedingten Ausführung mit `if` und `else if` können Sie ebenso über die mehrseitige Auswahl `switch` realisieren. Die Richtungsabfrage könnte folgendermaßen geschehen:



```
function scroll(laufweite,richtung) {
    switch (richtung) {
        case 0: // nach links
            ebene.style.left = (parseInt(ebene.style.left) - laufweite) + "px";
            break;
        case 1: // nach rechts
            ebene.style.left = (parseInt(ebene.style.left) + laufweite) + "px";
            break;
        case 2: // nach oben
            ebene.style.top = (parseInt(ebene.style.top) - laufweite) + "px";
            break;
        default: // nach unten
            ebene.style.top = (parseInt(ebene.style.top) + laufweite) + "px";
            break;
    }
}
```

9.6 Ebenen bewegen

Im vorherigen Kapitel haben Sie gelernt, wie Sie eine Ebene mit einem Mausklick an eine neue Position verschieben können. Klicken Sie sehr schnell auf die Pfeile der Grafik, können Sie eine Bewegung simulieren. Dies können Sie automatisch mit JavaScript realisieren.

Eine Bewegung erreichen Sie, indem Sie das Verschieben der Grafikebene in eine Schleife setzen. Dazu müssen Sie jedoch wissen, wie weit die Ebene in eine entsprechende Richtung verschoben werden soll. Übergeben Sie der Funktion `scroll` einen zusätzlichen Parameter, sobald auf einen Verweis auf der Grafik geklickt wird. Dieser Parameter beinhaltet den variablen Wert, der angibt an welcher Stelle die Bewegung stoppen soll.

```
<area shape="circle" coords="25,65,25" href="javascript:scroll(25,0,10)">
```

Damit die Funktion `scroll()` den Wert verarbeiten kann, müssen Sie die Parameterliste der Funktion erweitern. Speichern Sie z. B. den Wert in der Funktion in der Variablen `endposition`.

```
function scroll(laufweite, richtung, endposition)
```

In der Funktion erstellen Sie eine Zählschleife, die so lange durchlaufen wird, bis die Grafik die gewünschte Endposition erreicht hat. Dies realisieren Sie über eine `while`-Schleife.

Beispiel: Ebene verschieben (*kap09\bewegen_schnell.html*)

Die Funktion soll in diesem Beispiel anhand der Verschiebung der Ebene nach links verdeutlicht werden.

```
while(parseInt(ebene.style.left) > endposition) {
    ebene.style.left = (parseInt(ebene.style.left) - laufweite) + "px";
}
```

Die Ebene wird so lange um den Wert `laufweite` verschoben, wie die Ebenen-Koordinate `left` größer ist als die gewünschte Endposition.

Das Problem bei der Schleife ist, dass die Bewegung der Ebene zu schnell durchgeführt wird und somit für das menschliche Auge nicht sichtbar ist. Es wird eine Möglichkeit gesucht, das Ausführen der Bewegung zu verlangsamen. Die einzige Funktion, die dies möglich macht, ist die JavaScript-Funktion `setTimeout()`. Diese Funktion erlaubt jedoch nur das Aufrufen einer bestimmten Funktion. Dazu müssen Sie die `while`-Schleife wieder entfernen und stattdessen den Aufruf der `setTimeout()`-Funktion einsetzen.

```
function scroll(laufweite, richtung, endposition) {
    if (richtung==0) { // links
        ebene.style.left = (parseInt(ebene.style.left) - laufweite) + "px";
        if(parseInt(ebene.style.left) > endposition)
            setTimeout("scroll(\"" + laufweite + "," + richtung+"," + endposition+"")",20);
    }
}
```

Zusätzlich zum Positionieren der Ebene wird die derzeitige Position mit der gewünschten Endposition verglichen. Ist diese noch nicht erreicht, wird die Funktion `scroll()` erneut mit den Werten `laufweite`, `richtung` und `endposition` aufgerufen. Die Verzögerung der Bewegung, also die Wartezeit für den erneuten Funktionsaufruf, beträgt 20 Millisekunden.

Beachten Sie die Angabe der Variablen in der Funktion `setTimeout()`. Um deren Werte an die aufgerufene Funktion `scroll()` übergeben zu können, müssen Sie die ungewohnte Schreibweise mit dem Anführungszeichen `"` und dem Verkettungsoperator `,` beachten.

Eine fließende Bewegung erreichen Sie mit einer `laufweite` von 1 Pixel und einer Verzögerung von 10 Millisekunden in der `setTimeout()`-Funktion.



Beispiel: Ebene automatisch bewegen (`kap09\bewegen.html`)

Die komplette Funktion zum automatischen Bewegen einer Ebene sieht folgendermaßen aus:

```
function scroll(laufweite,richtung,endposition) {

    if (richtung==0) { // links
        ebene.style.left = (parseInt(ebene.style.left) - laufweite) + "px";
        if(parseInt(ebene.style.left) > endposition)
            setTimeout("scroll(\"+laufweite+\""+richtung+"," + endposition+"")",10);
    }

    else if (richtung==1) { // rechts
        ebene.style.left = (parseInt(ebene.style.left) + laufweite) + "px";
        if(parseInt(ebene.style.left) < endposition)
            setTimeout("scroll(\"+laufweite+\""+richtung+"," + endposition+"")",10);
    }

    else if (richtung==2) { // oben
        ebene.style.top = (parseInt(ebene.style.top) - laufweite) + "px";
        if(parseInt(ebene.style.top) > endposition)
            setTimeout("scroll(\"+laufweite+\""+richtung+"," + endposition+"")",10);
    }

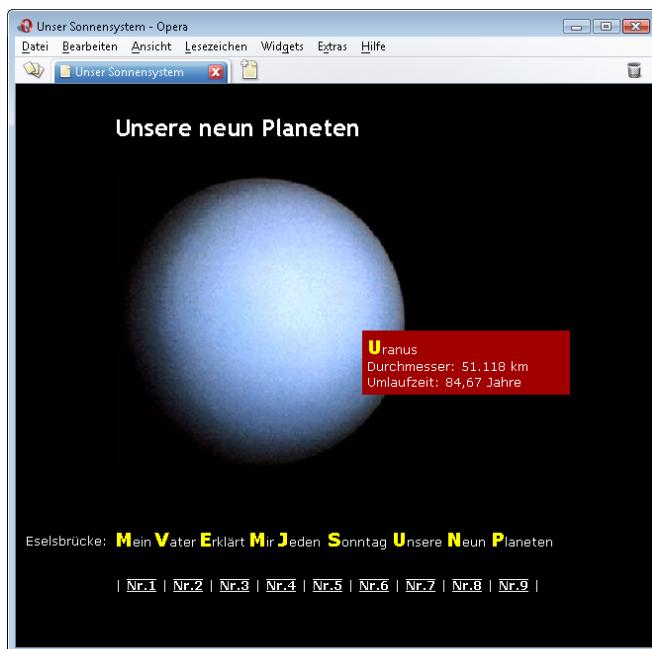
    else if (richtung==3) { // unten
        ebene.style.top = (parseInt(ebene.style.top) + laufweite) + "px";
        if(parseInt(ebene.style.top) < endposition)
            setTimeout("scroll(\"+laufweite+\""+richtung+"," + endposition+"")",10);
    }
}
```

9.7 Umfangreiches Beispiel

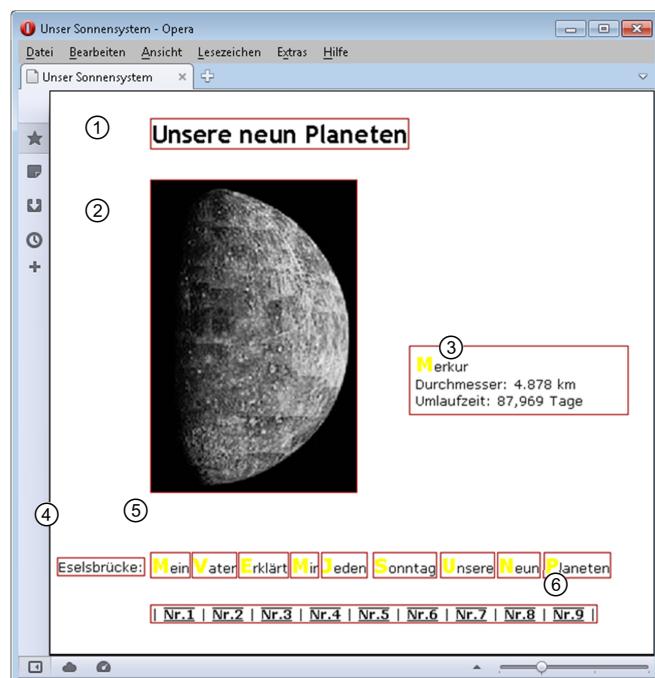
Im Beispiel soll zu jedem Planeten eine nähere Erläuterung zu seiner Größe und seiner Umlaufzeit um die Sonne angezeigt werden. Um sich die Reihenfolge der Planeten im Sonnensystem merken zu können, wird zu jedem Planeten ein Teil einer Eselsbrücke eingeblendet. Wurden alle Planeten aufgerufen, erscheint der komplette Satz mit den Wörtern, deren Anfangsbuchstaben denen der neun Planetennamen entsprechen.

Die nächste Abbildung zeigt das Grundlayout der Webseite. Um die einzelnen Informationen darzustellen, werden die verschiedenen Elemente in einzelnen Ebenen implementiert.

- ① Die Überschrift der Webseite bleibt immer am oberen Bildschirmrand sichtbar.
- ② An dieser Stelle sollen die einzelnen Planeten angezeigt werden. Nicht aktive Grafiken sollen unsichtbar dargestellt werden.
- ③ Diese Ebenen stellen zu jeder Grafik einen erläuternden Text zum Planeten dar. Auch sie sollen je nach Planet einzeln an- bzw. ausgeschaltet werden.
- ④ Der Text Eselsbrücke soll ständig sichtbar sein.
- ⑤ Jedes einzelne Wort der Eselsbrücke wird in einer einzelnen Ebene dargestellt und soll bei Aufruf eines Planeten animiert auf dem Bildschirm erscheinen.
- ⑥ Diese Ebenen enthalten die Verweise zum Abrufen der einzelnen Planeten. Hinter jedem Verweis verbirgt sich dabei ein Aufruf der JavaScript-Funktion zum Anzeigen des jeweiligen Planeten, der Informationsebene und des entsprechenden Teils der Eselsbrücke.



Informationen zum Sonnensystem in Opera



Die einzelnen sichtbaren Ebenen

Beispiel: CSS für die Planetensystem-Seite (*kap09\planeten\planeten.css*)

Zur Darstellung der einzelnen Ebenen werden die Formateigenschaften in einer separaten CSS-Datei angelegt.

```

① body      { font: small Verdana,Arial,sans-serif;
               color: #FFF; background-color:#000; }

② a         { font-weight: bold; color:#FFF; }
a:visited { text-decoration: overline underline; }
a:link    { text-decoration: overline underline; }
a:hover   { text-decoration: none; }

③ #header   { font: bold x-large 'Trebuchet MS',Verdana,Arial,sans-serif;
               position: absolute; left: 100px; top: 30px; }

④ .bildbox  { position: absolute; left: 100px; top: 90px;
               visibility: hidden; z-index: 5; }

⑤ .infobox  { position: absolute; left: 350px; top: 250px;
               visibility: hidden; z-index: 10; padding: 5px; width: 200px; }

⑥ .eselbox  { position: absolute; top: 350px; visibility: hidden; }

⑦ .navi     { position: absolute; left: 100px; top: 500px; }

⑧ .gb       { font: bold large Verdana,Arial,sans-serif; color: #FFFF00; }

```

- ① Für das gesamte Dokument wird die Schriftart Verdana in einer Schriftgröße von 10 pt festgelegt. Der Text der Webseite soll standardmäßig in weißer Farbe auf schwarzem Hintergrund dargestellt werden.
- ② Die Verweise werden oben und unten unterstrichen dargestellt (`text-decoration:underline overline`). Beim Überfahren mit der Maus soll der entsprechende Link keine Unterstreichung aufweisen.
- ③ Die Koordinaten und das Aussehen für die Darstellung der Überschrift mit der ID header werden festgelegt.
- ④ Die CSS-Klasse bildbox wird angelegt. Sie enthält die Definitionen für die Darstellung der Ebenen, die die Bilder der einzelnen Planeten anzeigen sollen. Nach dem Laden der Webseite sind dabei alle Grafik-Ebenen unsichtbar (`visibility:hidden`).
- ⑤ Die Klasse infobox enthält die Koordinaten und Formate für die Ebenen mit den Erläuterungen zu den einzelnen Planeten. Auch diese sind zu Beginn alle unsichtbar.
- ⑥ Die Klasse eselbox soll später den Ebenen mit den Teilen der Eselsbrücke zugewiesen werden. Hier wird auch nur der obere Abstand definiert, da jeder Teil der Eselsbrücke einen anderen linken Abstand haben wird.
- ⑦ Die Klasse navi enthält die Formatierung für die Navigation und demzufolge die Formatierungen der Verweise zum Aufruf der JavaScript-Funktionen.
- ⑧ Die Klasse gb dient dazu, den ersten Buchstaben eines jeden Planeten vergrößert darzustellen.

Beispiel: Planetensystem-Seite (`kap09\planeten\planeten.html`)

Als Nächstes erstellen Sie den HTML-Code für die Webseite mit den einzelnen Ebenen. Die Überschrift und die ersten neun Ebenen, welche die Grafiken der Planeten enthalten, werden als Erstes angegeben.

```

<html>
<head>
  <meta charset="utf-8">
  <title>Unser Sonnensystem</title>
  ① <link type="text/css" rel="stylesheet" href="planeten.css">
</head>
<body>
  ② <div id="header">Unsere neun Planeten</div>

```

```
(3) <!-- Planetenbilder-->
    <div id="planetenbild1" class="bildbox">
        </div>
    <div id="planetenbild2" class="bildbox">
        </div>
    <div id="planetenbild3" class="bildbox">
        </div>
    <div id="planetenbild4" class="bildbox">
        </div>
    <div id="planetenbild5" class="bildbox">
        </div>
    <div id="planetenbild6" class="bildbox">
        </div>
    <div id="planetenbild7" class="bildbox">
        </div>
    <div id="planetenbild8" class="bildbox">
        </div>
    <div id="planetenbild9" class="bildbox">
        </div>
```

- ① Zur Formatierung der einzelnen HTML-Tags binden Sie die eben angelegte CSS-Datei *planeten.css* ein.
- ② Die Ebene, die als Überschrift fungieren soll, erhält aufgrund der ID *header* automatisch die Stylesheet-Definitionen und wird im Browser 100 Pixel von links und 30 Pixel von der oberen Begrenzung des Browsers entfernt dargestellt.
- ③ Jede einzelne Planetengrafik erhält die Formatierung der Klasse *bildbox* und wird dadurch an der Koordinate 100, 90 unsichtbar eingefügt. Der festgelegte Name *planetenbild1-9* ermöglicht den späteren Zugriff auf die Eigenschaften jeder einzelnen Ebene.

Im nächsten Schritt legen Sie die Ebenen mit den Informationen zu den einzelnen Planeten fest.

```
(4) <!-- Informationen zu den einzelnen Planeten-->
    <div id="planeteninfo1" class="infobox">
        <span class="gb">M</span>erkur<br>Durchmesser: 4.878 km<br>
        Umlaufzeit: 87,969 Tage</div>
    <div id="planeteninfo2" class="infobox">
        <span class="gb">V</span>enus<br>Durchmesser: 12.104 km<br>
        Umlaufzeit: 224,70 Tage</div>
    <div id="planeteninfo3" class="infobox">
        <span class="gb">E</span>rde<br>Durchmesser: 12.742 km<br>
        Umlaufzeit: 365,2 Tage</div>
    <div id="planeteninfo4" class="infobox">
        <span class="gb">M</span>ars<br>Durchmesser: 6.794 km<br>
        Umlaufzeit: 686,98 Tage</div>
    <div id="planeteninfo5" class="infobox">
        <span class="gb">J</span>upiter<br>Durchmesser: 142.796 km<br>
        Umlaufzeit: 11,869 Jahre</div>
    <div id="planeteninfo6" class="infobox">
        <span class="gb">S</span>aturn<br>Durchmesser: 120.000 km<br>
        Umlaufzeit: 29,46 Jahre</div>
    <div id="planeteninfo7" class="infobox">
        <span class="gb">U</span>ranus<br>Durchmesser: 51.118 km<br>
        Umlaufzeit: 84,67 Jahre</div>
    <div id="planeteninfo8" class="infobox">
        <span class="gb">N</span>eptun<br>Durchmesser: 49.424 km<br>
        Umlaufzeit: 165,49 Jahre</div>
    <div id="planeteninfo9" class="infobox">
        <span class="gb">P</span>luto<br>Durchmesser: 2.300 km<br>
        Umlaufzeit: 247,7 Jahre</div>
```

- ④ Alle Ebenen erhalten die Stylesheets der definierten CSS-Klasse `infobox`. Dadurch werden die Informationen rechts neben den einzelnen Grafiken an der Koordinate 350, 250 angezeigt und zu Beginn unsichtbar gesetzt. Mit der Vergabe der ID `planeteninfo1-9` können diese Elemente im Nachhinein einzeln eingeblendet werden.
- ⑤ Die Zuweisung der CSS-Klasse `gb` zu Beginn eines jeden Planetennamens lässt den ersten Buchstaben etwas größer erscheinen.

Die effektvollsten Ebenen, die nach dem Klick auf einen Verweis an ihre Endposition verschoben werden sollen, werden über die Klasse `eselbox` definiert. Bisher wurden die Ebenen der Grafiken und der Erläuterungen immer an derselben Stelle positioniert. Nun sollen jedoch die Ebenen nebeneinanderstehen. Dazu werden die Definitionen der linken und der oberen Ebenenkoordinaten direkt am Element festgelegt. Die Angabe von `top: 350` bewirkt, dass die Ebenen der Eselsbrücke unterhalb der Grafikebenen positioniert werden. Später sollen sie hinter den anderen Ebenen hervorkommen und sich an die endgültige Position bewegen.

```
<!-- Eselsbrücken -->
⑥ <div style="position:absolute; left:10px; top:455px;
      z-index:20;">Eselsbrücke:</div>
⑦ <div id="esel1" class="eselbox" style="left:100px;">
      <span class="gb">M</span>ein</div>
    <div id="esel2" class="eselbox" style="left:140px;">
      <span class="gb">V</span>ater</div>
    <div id="esel3" class="eselbox" style="left:185px;">
      <span class="gb">E</span>rklärt</div>
    <div id="esel4" class="eselbox" style="left:235px;">
      <span class="gb">M</span>ir</div>
    <div id="esel5" class="eselbox" style="left:265px;">
      <span class="gb">J</span>eden</div>
    <div id="esel6" class="eselbox" style="left:315px;">
      <span class="gb">S</span>ontag</div>
    <div id="esel7" class="eselbox" style="left:380px;">
      <span class="gb">U</span>nser</div>
    <div id="esel8" class="eselbox" style="left:435px;">
      <span class="gb">N</span>eun</div>
    <div id="esel9" class="eselbox" style="left:480px;">
      <span class="gb">P</span>laneten</div>
```

- ⑥ Der Text `Eselsbrücke` wird an die Koordinaten-Position 10, 455 gesetzt und bleibt dort auch stehen. Diese Ebene wird im weiteren Verlauf nicht verschoben.
- ⑦ Jede einzelne Ebene erhält entsprechend den neun Planeten den Namen `esel1-9`. Durch die Zuweisung der CSS-Klasse `eselbox` wird sichergestellt, dass sie standardmäßig 350 Pixel vom oberen Rand entfernt platziert werden. Die seitliche Positionierung wird direkt am Element mit der Eigenschaft `left` festgelegt. Wie bei den Informationen zu den einzelnen Planeten wird auch hier der erste Buchstabe mithilfe der Klasse `gb` größer formatiert.

Als Letztes legen Sie die Verweise an, welche die Ereignisse zum Bewegen und Einblenden der verschiedenen Ebenen auslösen sollen.

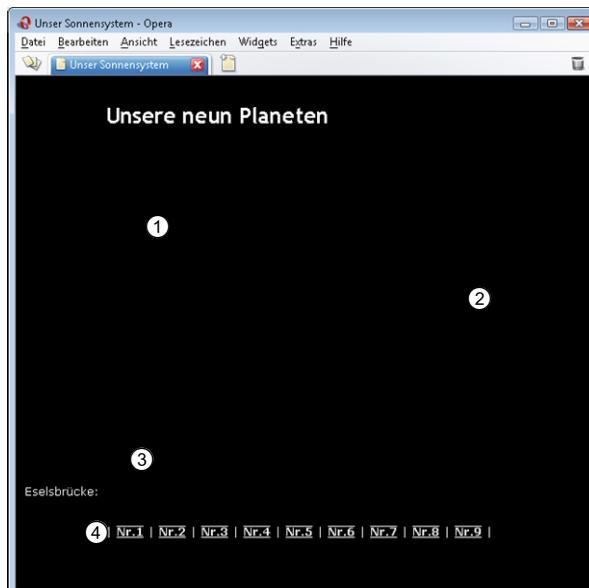
```

<!--Links-->
⑧ <div class="navi">
⑨ | <a href="#" onClick="umschalt('1'); return false;">Nr.1</a>
| <a href="#" onClick="umschalt('2'); return false;">Nr.2</a>
| <a href="#" onClick="umschalt('3'); return false;">Nr.3</a>
| <a href="#" onClick="umschalt('4'); return false;">Nr.4</a>
| <a href="#" onClick="umschalt('5'); return false;">Nr.5</a>
| <a href="#" onClick="umschalt('6'); return false;">Nr.6</a>
| <a href="#" onClick="umschalt('7'); return false;">Nr.7</a>
| <a href="#" onClick="umschalt('8'); return false;">Nr.8</a>
| <a href="#" onClick="umschalt('9'); return false;">Nr.9</a> |
</div>
</body>
</html>

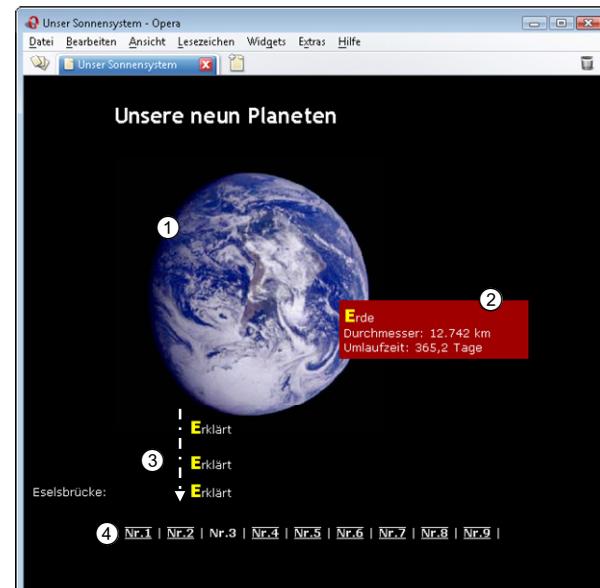
```

- ⑧ Die Hyperlinks werden über das Tag `<div>` als Block markiert und über die Klasse `navi` entsprechend ausgerichtet.
- ⑨ Über die Ereignisabfrage `onClick` wird die JavaScript-Funktion `umschalt()` aufgerufen, sobald ein Hyperlink ausgelöst wird. Die Namen der Ebenen für die Grafiken, für die Informationen und für die Eselsbrücken enden immer auf eine Zahl von 1 bis 9, entsprechend der Reihenfolge der Planeten in unserem Sonnensystem. Um jeweils die drei Ebenen, die verändert werden sollen, ansprechen zu können, reicht es, der Funktion jeweils die Nummer des gewählten Planeten zu übergeben. Mit `return false` wird festgelegt, dass der Verweis beim Auswählen nicht ausgelöst werden soll. Somit wird nicht auf eine andere Seite verzweigt.

Bisher haben Sie das Aussehen der Webseite festgelegt. Was nun noch fehlt, ist der JavaScript-Code, das Herzstück der Webseite.



Die Ebenen sind nicht sichtbar



Auslösen des Verweises und Anzeige der Informationen

Bei einem Klick auf einen Verweis ④ sollen die entsprechende Grafik ① und die Informationsebene ② eingeblendet werden. Zusätzlich soll der jeweilige Teil der Eselsbrücke ③ an seine Endposition verschoben werden.

Fügen Sie dazu den ersten Teil des JavaScripts in den Kopfbereich des HTML-Dokuments ein:

```

① <script>
②   var letztesBild = 'planetenbild1';
③   var letzteBox  = 'planeteninfo1';

④   function umschalt(i) {
⑤     EbeneEsel = 'esel'+i;
⑥     EbeneBild = 'planetenbild'+i;
⑦     EbeneBox  = 'planeteninfo'+i;

⑧     ebene=document.getElementById(EbeneEsel).style;
⑨     ebene.top=350 + "px";

⑩     letztesBild = document.getElementById(letztesBild).style;
⑪     jetzigesBild= document.getElementById(EbeneBild).style;
⑫     letzteBox   = document.getElementById(letzteBox).style;
⑬     jetzigeBox  = document.getElementById(EbeneBox).style;

⑭     letztesBild.visibility = 'hidden';
⑮     letzteBox.visibility  = 'hidden';
⑯     jetzigesBild.visibility = 'visible';
⑰     jetzigeBox.visibility  = 'visible';

⑱     letztesBild = EbeneBild;
⑲     letzteBox   = EbeneBox;

⑳     bewegung(i);
}

```

- ① Jede Ebene, die angezeigt werden soll, muss auch wieder unsichtbar werden. Dazu muss der Name dieser Ebene in einer Variablen gespeichert werden. Die Variable `letztesBild` enthält den Namen der Grafikebene und `letzteBox` den Namen der Informationsebene, die zuletzt angezeigt wurden. Zu Beginn werden ihnen die jeweils ersten Ebenen zugewiesen (`planetenbild1` und `planetenbox1`).
- ② Der Name der Ebene für die aktive Eselsbrücke wird aus der Zeichenkette `esel` und der Variablen `i`, die der Funktion `umschalt()` beim Aufruf übergeben wurde, zusammengesetzt. Die Namen der Ebenen für die Grafik und die Information werden ebenfalls aus den entsprechenden Zeichenketten und der Variablen `i` gebildet.
- ③ Damit im weiteren Verlauf einfacher auf die CSS-Eigenschaften des jeweiligen Elements zugegriffen werden kann, wird das Objekt in der Variablen `ebene` gespeichert. Danach wird der Ebene mit der Eselsbrücke eine obere Position von 350 Pixeln zugewiesen.
- ④ In den vier Variablen werden die entsprechenden Objekte der letzten und der gerade aktiven Grafik sowie der letzten und der aktiven Informationsebene gespeichert.
- ⑤ Die letzte angezeigte Grafik und die entsprechende Infobox werden unsichtbar gesetzt (`visibility=hidden`). Die Elemente des neu gewählten Planeten werden sichtbar gemacht (`visible`).
- ⑥ Damit bei der nächsten Auswahl eines Planeten die jetzt aktiven Elemente unsichtbar gemacht werden, müssen deren Bezeichner den beiden Variablen `letztesBild` und `letzteBox` übergeben werden.
- ⑦ Um die Ebene der Eselsbrücke in Bewegung zu versetzen, wird die gleichnamige Funktion `bewegung()` mit der Variablen `i`, die den Wert der aktuellen Ebene enthält, aufgerufen.

```

function bewegung(i) {
    ziel=450;
    EbeneEsel='esel'+i;
    ebene=document.getElementById(EbeneEsel).style;
    ebene.visibility = 'visible';

    if(parseInt(ebene.top) < ziel) {
        ebene.top = (parseInt(ebene.top)+1) + "px";
        setTimeout('bewegung('+i+')',5);
    }
}
</script>

```

- ⑧ Die Variable `ziel` enthält die y-Koordinate, zu der die Ebene bewegt werden soll. Ziel wäre somit die y-Koordinate 450 Pixel.
- ⑨ Wie in der Funktion `umschalt()` wird der Name der Ebene für die aktive Eselsbrücke aus der Zeichenkette `esel` und der Variablen `i` zusammengesetzt, und die verschiedenen Objektaufrufe werden in der Variablen `ebene` gespeichert.
- ⑩ Die anzuzeigende Ebene wird sichtbar gemacht.
- ⑪ Für die Berechnung der Ebenenbewegung wird die aktuelle Position `top` ausgelesen. Mit der Angabe von `parseInt` wird die ermittelte Zeichenkette in eine Zahl umgewandelt. Ist der Wert der Variablen `ziel` noch nicht erreicht, d. h., ist die Position von 450 Pixeln noch nicht erreicht, wird in die `if`-Auswahl gesprungen. Hat die aktuelle Ebene die Zielposition erreicht, wird die Funktion `bewegung()` verlassen.
- ⑫ Die Eigenschaft `top` der aktuellen Ebene wird um den Wert 1 erhöht. Die Maßeinheit `px` wird als Zeichenkette angefügt. Dies bedeutet, die aktuelle Ebene bewegt sich um einen Pixel nach unten.
- ⑬ Um die Bewegung bis zur Zielkoordinaten durchzuführen, muss die Funktion `bewegung()` erneut aufgerufen werden. Dies geschieht über die JavaScript-Funktion `setTimeout()`, der Sie den aktuellen Wert des Ebenenzählers `i` übergeben. Zusätzlich geben Sie die Verzögerung, mit der sich die Ebene bewegen soll, in Millisekunden an. Je höher die Verzögerung, desto langsamer die Bewegung.

Das Ergebnis ist eine Webseite, die komplett aus einzelnen Ebenen besteht, die per JavaScript ein- und ausgeschaltet sowie bewegt werden können. Kurz gesagt, Ihre Webseite ist eine DHTML-Webseite.

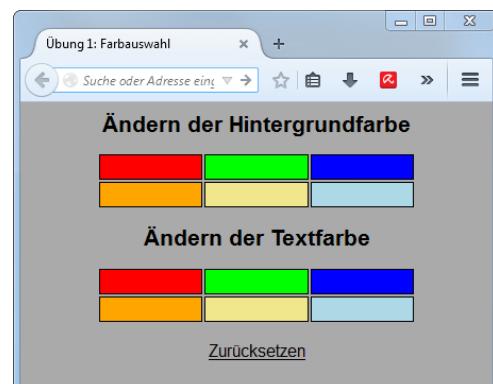
9.8 Übungen

Übung 1: Elemente einfärben

Übungsdatei: --

Ergebnisdatei: uebung1.html

1. Erstellen Sie eine Tabelle mit verschiedenenfarbigen Zellhintergründen. Bei einem Klick auf eine Zelle soll diese Farbe als Hintergrundfarbe der Webseite verwendet werden. Eine zweite Tabelle soll für die Farbe des Dokumententextes verwendet werden.



Auswählen der verschiedenen Farben in Firefox

Übung 2: Verweise mit eingeblendeten Erklärungen

Übungsdatei: --

Ergebnisdatei: uebung2.html

1. Gestalten Sie eine Liste mit verschiedenen Verweisen. Beim Überfahren mit der Maus soll eine nähere Erläuterung zu den verlinkten Webseiten dargestellt werden.

Übung 2: Hinweise bei Verweisen



Übung 3: Eine interaktive Speisekarte

Übungsdatei: --

Ergebnisdatei: uebung3.html

1. Entwerfen Sie eine Webseite für ein Restaurant. Dabei soll der Besucher über eine Liste auswählen können, welche speziellen Gerichte angezeigt werden sollen. Das Einblenden der Speisen soll über verschiedene Ebenen erfolgen.

Vorspeisen	
Zaziki Joghurt mit Gurken und Knoblauch	2,75 €
Taramas griechische Kaviarcreme	3,00 €
Htipiti Schafskäsecrème	3,75 €
Htapodaki Oktopussalat	4,75 €

Die interaktive Speisekarte

10 JavaScript-APIs der HTML5-Spezifikation

In diesem Kapitel erfahren Sie

- ✓ Wie Sie APIs nutzen, um die Funktionalität Ihrer Webseiten zu erweitern
- ✓ Wie Sie mit der Geolocation-API arbeiten
- ✓ Wie Sie den Canvas-2D-Kontext einsetzen

Voraussetzungen

- ✓ Fortgeschrittene HTML-Kenntnisse
- ✓ Kenntnisse der Funktionsweise von JavaScript

10.1 Was ist eine API?

Bei einer **API** handelt es sich um eine Sammlung von Programmieranweisungen und Standards für den Zugriff auf eine Software-Anwendung.



API ist die Abkürzung von **Application Programming Interface** (Anwendungsprogrammierschnittstelle).

Hauptbestandteile der HTML-Spezifikation sind zahlreiche JavaScript-APIs. Sie bieten dem Programmierer die Möglichkeit, in HTML voll funktionsfähige Webanwendungen zu entwickeln und gehören damit zu den wichtigsten und zukunftsweisenden Features von HTML5.

10.2 Häufig verwendete JavaScript-APIs

Die HTML5-Spezifikation bietet viele JavaScript-APIs. Einige häufig verwendete JavaScript-APIs werden im Folgenden vorgestellt:

Drag & Drop	Die Drag-and-Drop-API ermöglicht es Ihnen, ein Objekt auf einer Webseite zu ziehen und an einer anderen Seite abzulegen. Jedes beliebige Element kann ziehbar sein.
Local Storage	Mit Local Storage können Daten lokal im Browser des Benutzers gespeichert werden. Diese Technologie kann somit als Ersatz für Cookies dienen. Die Speicherkapazität liegt mit mindestens 5 MB auch deutlich höher als bei Cookies.
Application Cache	Mit Application Cache können beliebige, durch eine sogenannte Manifest-Datei spezifizierte Ressourcen einer Webseite im Anwendungscache gespeichert werden. Sie sind anschließend auch offline verfügbar. Mit der weiten Verbreitung mobiler Geräte wie etwa Tablets wird diese Technologie immer wichtiger.
Web Workers	Web Workers delegieren die Ausführung von JavaScript an sogenannte Hintergrund-Threads. Das heißt, dass interaktive Skripts, die auf Benutzereingaben reagieren, dann nicht mehr ausgebremst werden, sondern parallel laufen. Webworkers sind vor allem für komplexe Webanwendungen sinnvoll.
Server Sent Events	Server Sent Events öffnen eine HTTP-Verbindung und senden Ereignisse als DOM-Events direkt an den Client.

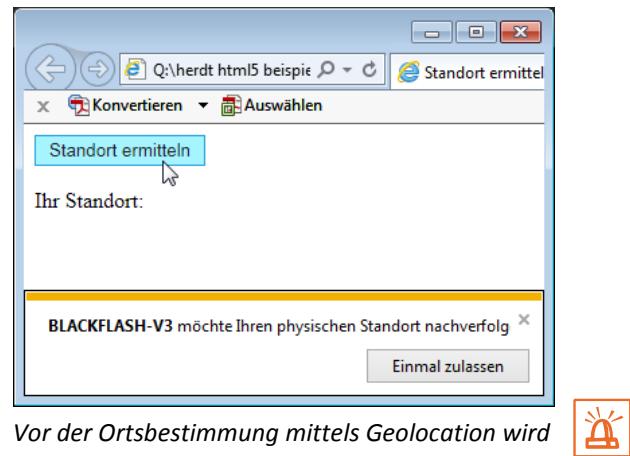
Geolocation	Mit der Geolocation-API (vgl. Abschnitt 9.3) kann die geografische Position des Nutzers bestimmt werden. Die API gibt Längen- und Breitengrad zurück. Dadurch wird es beispielsweise möglich, dem jeweiligen Nutzer individuelle, zu den Koordinaten passende Dienste anzubieten (LBS = Location Based Services).
Canvas-2D-Kontext	Die Canvas-2D-Kontext-API (vgl. Abschnitt 9.4) wird mit dem HTML5-canvas-Element verwendet, um grafische Elemente darzustellen.

10.3 Die Geolocation-API

Sie verwenden die Geolocation-API, um auf den Standort eines Nutzers zuzugreifen. So ist es beispielsweise möglich, dem Nutzer interessante Orte in seiner Nähe zu zeigen oder Navigationsanwendungen für mobile Geräte zu entwickeln.

Die Ortsbestimmung wird aufgrund von IP-Adressen, web-basierten Datenbanken, drahtlosen Netzwerkverbindungen und GPS-Technologien durchgeführt. Von der verwendeten Technologie hängt auch die Genauigkeit der Standort-ermittlung ab. Die größte Genauigkeit ermöglichen Geräte mit **GPS** (z. B. iPhones).

Da die Ortsbestimmung die Privatsphäre der Nutzer gefährden kann, steht die Position erst nach Genehmigung durch den Nutzer zur Verfügung.



Vor der Ortsbestimmung mittels Geolocation wird stets die Genehmigung des Nutzers eingeholt.



Funktionsweise der Geolocation-API

Die Geolocation-API basiert auf der Eigenschaft `geolocation` des globalen JavaScript-Objekts `navigator`.

Das `navigator`-Objekt liefert Informationen über den Browser und das System des Nutzers. Per JavaScript können mit `navigator.geolocation` die Standortinformationen des Geräts des Nutzers bestimmt werden. Sobald die Ortsinformationen abgerufen sind, wird ein Positionsobjekt erstellt und mit den Daten gefüllt.

Die vollständige Spezifikation der Geolocation-API finden Sie beim W3C unter <http://dev.w3.org/geo/api/spec-source.html>.



Methoden des `geolocation`-Objekts

Das Objekt `navigator.geolocation` hat drei Methoden:

<code>getCurrentPosition (successCallback, errorCallback, options)</code>	Ruft die aktuelle geografische Position des Benutzers einmalig ab. <ul style="list-style-type: none"> ✓ Argument <code>successCallback</code> (erforderlich): Das Callback mit der aktuellen Position ✓ Argument <code>errorCallback</code> (optional): Das Callback für den Fehlerfall ✓ Argument <code>options</code> (optional): Die Geolocation-Optionen - möglich sind <code>enableHighAccuracy</code> (für den Fall, dass das Gerät des Benutzers eine hohe Genauigkeit ermöglicht), <code>timeout</code> (die maximal mögliche Dauer vom Aufruf bis zum <code>successCallback</code> in Millisekunden) und <code>maximumAge</code> (wie lange ein im Cache gespeicherter Standort verwendet werden darf).
---	---

<code>watchPosition(successCallback, errorCallback, options)</code>	Ruft die Position des Benutzers ab und aktualisiert sie, während der Nutzer sich bewegt. Die Argumente entsprechen denen der Methode <code>getCurrentPosition</code> . Nur sinnvoll für die Verwendung mit mobilen Geräten wie etwa Tablets oder Smartphones
<code>clearWatch(watchID)</code>	Beendet die Methode <code>watchPosition()</code> .

Die Eigenschaften des von der Methode `getCurrentPosition()` zurückgegebenen Positionsobjekts

Die Methode `getCurrentPosition()` gibt im Erfolgsfall ein Positionsobjekt mit dem Längen- und Breitengrad und der Genauigkeit dieser Angaben sowie ggf. weiteren Eigenschaften zurück.

Die Eigenschaften des Positionsobjekts:

<code>coords.latitude</code>	Der aktuelle (geschätzte) Breitengrad
<code>coords.longitude</code>	Der aktuelle (geschätzte) Längengrad
<code>coords.accuracy</code>	Die Genauigkeit der Ortsbestimmung in Metern
<code>coords.altitude</code>	Die geschätzte Höhe über dem Meeresspiegel in Metern
<code>coords.altitudeAccuracy</code>	Die Genauigkeit der Höhenangabe in Metern
<code>coords.heading</code>	Der Richtungswinkel im Uhrzeigersinn von Norden aus
<code>coords.speed</code>	Die Geschwindigkeit in Metern pro Sekunde
<code>timestamp</code>	Datum und Uhrzeit der erzeugten Daten



Alle Eigenschaften außer `coords.latitude`, `coords.longitude` und `coords.accuracy` sind optional und geben `null` zurück, wenn Gerät oder Server diese nicht ermitteln können.

Beispiel: Die geografische Position des Nutzers mit `getCurrentPosition()` ermitteln ([kap10/geolocation.html](#))

Das folgende Beispiel liefert den Längen- und Breitengrad der Position des Benutzers zurück:

<code><body></code>	
<code><button onclick="standortErmitteln()">Standort ermitteln</button></code>	
<code><p>Ihr Standort:</p></code>	
<code><p id="ausgabe"></p></code>	
<code><script></code>	
<code>var x = document.getElementById("ausgabe");</code>	
<code>① function standortErmitteln() {</code>	
<code> if (navigator.geolocation) {</code>	
<code> navigator.geolocation.getCurrentPosition(zeigeStandort);</code>	
<code> } else {</code>	
<code> } else {</code>	
<code> x.innerHTML = "Ihr Browser unterstützt keine Standortermittlung.";</code>	
<code> }</code>	
<code>④ }</code>	
<code>function zeigeStandort(standort) {</code>	
<code> x.innerHTML = "Breitengrad: " + standort.coords.latitude +</code>	
<code> "
Längengrad: " + standort.coords.longitude;</code>	
<code>}</code>	
<code></script></code>	
<code></body></code>	

Die geografische Position eines Nutzers ermitteln

- ① In der Funktion `standortErmitteln` wird ermittelt, ob der Browser des Nutzers Geolocation unterstützt.
- ② Ist dies der Fall, wird die Methode `getCurrentPosition()` ausgeführt und ein Koordinatenobjekt an die Funktion `zeigeStandort` übergeben.
- ③ Andernfalls erhält der Nutzer eine entsprechende Meldung.
- ④ Die Funktion `zeigeStandort()` zeigt Breiten- und Längengrad an.

Standort ermitteln
Ihr Standort:
Breitengrad: 48.6616037
Längengrad: 9.3501336

Den ungefähren Standort des Benutzers ermitteln und ausgeben

Mit den Standardeinstellungen des Internet Explorers müssen Sie zuerst explizit bestätigen, dass der Browser Ihren Standort abfragen darf, damit dieses und die übrigen Geolocation-Beispiele funktionieren.



Beispiel: Den Standort des Benutzers in einer Google-Maps-Karte anzeigen (kap10/geolocation-googlemaps.html)

Das Ergebnis der Standortermittlung lässt sich im Browser in einer Karte anzeigen. Dazu müssen Sie auf einen Online-Kartendienst zugreifen, der Längen- und Breitengrad nutzen kann. Ein Beispiel dafür ist Google Maps.

```

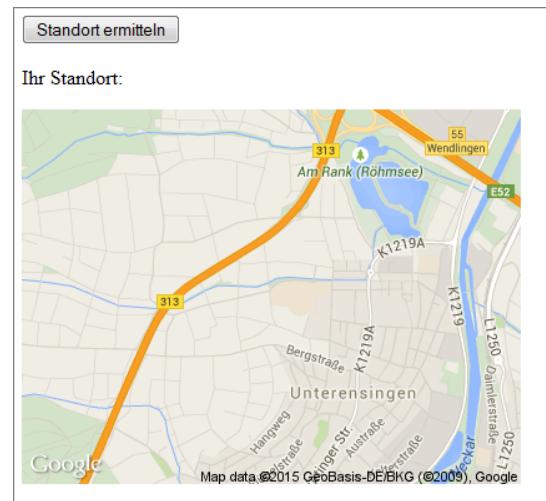
<body>
<button onclick="standortErmitteln()">Standort ermitteln</button>
<p>Ihr Standort:</p>
<div id="karte"></div>
<script>
var x = document.getElementById("ausgabe");
function standortErmitteln() {
    if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(zeigeStandort);
    } else {
        x.innerHTML = "Ihr Browser unterstützt die Ermittlung des Standorts nicht.";
    }
}
① function zeigeStandort(standort) {
    var latlon = standort.coords.latitude + "," + standort.coords.longitude;
    var kartenbild = "http://maps.googleapis.com/maps/api/staticmap?center=" +
        latlon+"&zoom=14&size=400x300&sensor=false";
    document.getElementById("karte").innerHTML = "<img src='"+kartenbild+"'>";
}
</script>
</body>

```

Standort in Google Maps anzeigen

- ① Im Beispiel werden in der Funktion `zeigeStandort` die Koordinaten in die Variable `breiteLaenge` geschrieben. Die Variable `breiteLaenge` wird mit der Google Static Maps-API verwendet, um das Kartenbild auf der Webseite auszugeben.

 Mit der Google Static Maps-API können Sie ein Google Maps-Bild auf Ihrer Webseite anzeigen. Weitere Informationen über die API erhalten Sie unter <https://developers.google.com/maps/documentation/staticmaps>.



Den Standort des Benutzers in einer Google-Maps-Karte ausgeben

Fehlerbehandlung

Der zweite Parameter der Methode `getCurrentPosition()` oder `watchPosition()` wird verwendet, um Fehler zu behandeln.

Wenn der Aufruf der Geolocation-API einen Fehler verursacht, wird die `errorCallback`-Rückruffunktion mit einem neuen `positionError`-Objekt aufgerufen. Das Objekt gibt die Art des Fehlers an.

Das `positionError`-Objekt besitzt die Eigenschaften `code` und `message`.

<code>code</code>	Gibt eine Ganzzahl zurück, die die Art des Fehlers spezifiziert.
<code>message</code>	Enthält eine Fehlermeldung.

Mögliche Fehlercodes:

<code>positionError.PERMISSION_DENIED</code> (1)	Die Website ist nicht berechtigt, Geolocation zu verwenden.
<code>positionError.POSITION_UNAVAILABLE</code> (2)	Der aktuelle Standort kann nicht ermittelt werden.
<code>positionError.TIMEOUT</code> (3)	Der aktuelle Standort kann in der angegebenen Zeit nicht ermittelt werden.

Das folgende Listing zeigt eine Standortbestimmung mit implementierter Fehlerbehandlung. Die Fehlermeldung wird in diesem Fall mit der Eigenschaft `innerHTML` direkt in die Webseite eingefügt.

```

function showError(error) {
    switch(error.code) {
        case error.PERMISSION_DENIED:
            x.innerHTML = "Zugriff auf den Standort des Benutzers verweigert."
            break;
        case error.POSITION_UNAVAILABLE:
            x.innerHTML = "Standortinformation ist nicht verfügbar."
            break;
        case error.TIMEOUT:
            x.innerHTML = "Zeitüberschreitung der Anfrage."
            break;
        case error.UNKNOWN_ERROR:
            x.innerHTML = "Ein unbekannter Fehler ist aufgetreten."
            break;
    }
}

```

Fehlerbehandlung für Geolocation

10.4 Der Canvas-2D-Kontext

Das HTML5-Element `canvas` stellt APIs bereit, die es möglich machen, mit einer Skriptsprache (normalerweise JavaScript) dynamisch Bilder und Animationen auf einer Webseite zu erzeugen. Mit dem `canvas`-Element selbst können keine Grafiken erstellt werden. Es dient lediglich als Container für die mit einer Skriptsprache erzeugten Grafiken.

Es gibt mehrere Methoden zum Zeichnen von Pfaden, Rechtecken, Kreisen und Text, zum Ändern der Farben sowie zum Hinzufügen von Bildern. Außerdem ist es möglich, Objekte auf dem Canvas zu animieren und sie auf Nutzeraktionen wie etwa Mausklick, Tastendruck und Ähnliches reagieren zu lassen. So lassen sich mithilfe des `canvas`-Elements unter anderem JavaScript-Spiele direkt in die Webseite einbinden.

Die meisten modernen Browser unterstützen das `canvas`-Element. Ab folgender Browser-Version kann das `canvas`-Element angezeigt werden:

- | | | |
|-----------------------|-------------------------|-------------|
| ✓ Google Chrome 4.0 | ✓ Safari 3.1 | ✓ Opera 9.0 |
| ✓ Mozilla Firefox 2.0 | ✓ Internet Explorer 9.0 | |



Ein `canvas`-Element in die Seite einfügen

Damit Sie per JavaScript grafische Elemente erstellen können, fügen Sie zunächst ein HTML-`canvas`-Element in Ihre Webseite ein.

- ▶ Erzeugen Sie ein HTML-Dokument.
- ▶ Geben Sie im `<body>`-Bereich das `canvas`-Element ein.

```
<canvas id="meinCanvas" width="500" height="300"></canvas>
```

- ✓ Die Angabe einer `id` ist notwendig, damit Sie das `canvas`-Element in JavaScript auswählen können.
- ✓ `width` bestimmt die Breite des Canvas in Pixeln.
- ✓ `height` ist die Höhe des Canvas in Pixeln.

Soll der Canvas eine Formatierung wie etwa einen Rahmen erhalten, weisen Sie die Formatierung per CSS zu.



Formen und Linien auf den Canvas zeichnen

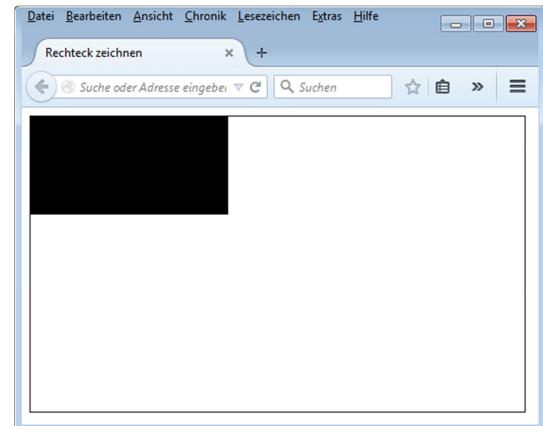
Um per JavaScript Formen zu zeichnen, nutzen Sie die Koordinaten des Canvas: Das Canvas-Raster beginnt in der linken oberen Ecke mit den Koordinaten 0,0.

Beispiel: Ein Rechteck auf den Canvas zeichnen (*kap10\canvas-rechteck.html*)

```
<html>
<head>
<meta charset="utf-8" >
<title>Rechteck zeichnen</title>
</head>
<body>
<canvas id="meinCanvas" width="500" height="300"></canvas>
<script>
① var canvas = document.getElementById("meinCanvas");
② var ctx = canvas.getContext("2d");
③ ctx.fillRect(0, 0, 200, 100);
</script>
</body>
</html>
```

Gefülltes Rechteck

- ① Die Methode `getElementById` gibt ein DOM-Objekt zurück, welches das Element mit der jeweiligen ID darstellt.
- ② Aus dem `canvas`-Element wird der Zeichenbereich erzeugt, ein JavaScript-Objekt mit allen Methoden und Eigenschaften für das Zeichnen auf dem Canvas. Das Argument `"2d"` vermittelt, dass auf dem Canvas eine zweidimensionale Grafik gezeichnet werden soll.
- ③ Die Methode `fillRect` erzeugt ein gefülltes Rechteck auf dem Canvas. Wenn nichts anderes angegeben wird, ist die Füllfarbe schwarz.
 - ✓ Das erste und zweite Argument der Methode `fillRect` sind die x- und y-Koordinaten der linken oberen Ecke des Rechtecks.
 - ✓ Das dritte Argument ist die Breite des Rechtecks.
 - ✓ Das vierte Argument ist die Höhe des Rechtecks.



200 Pixel breites und 100 Pixel hohes gefülltes Rechteck auf dem Canvas

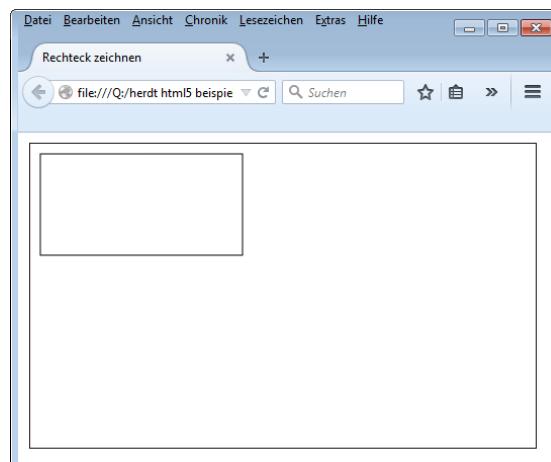
Ein ungefülltes Rechteck zeichnen

Für ungefüllte Rechtecke gibt es die Methode `strokeRect`, die analog zur Methode `fillRect` funktioniert.

```
var canvas =  
    document.getElementById("meinCanvas");  
var ctx = canvas.getContext("2d");  
① ctx.lineWidth = 4;  
② ctx.strokeRect(10, 10, 200, 100);
```

Ungefülltes Rechteck

- ① Mit `lineWidth` legen Sie die Breite der Kontur in Pixeln fest.
- ② Das ungefüllte Rechteck zeichnen Sie mit der Methode `strokeRect`. Die Argumente für diese Methoden sind dieselben wie bei `fillRect`.



Ein ungefülltes Rechteck erzeugen

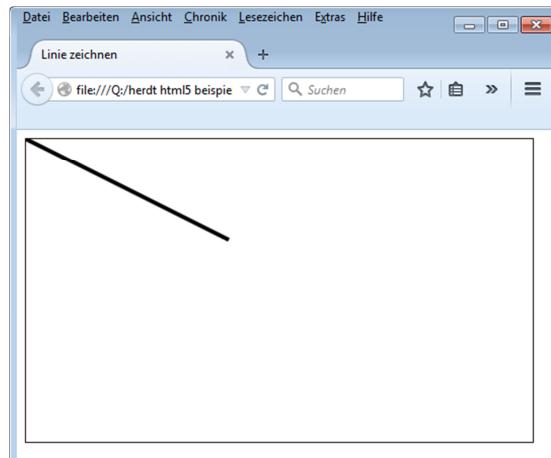
Gerade Pfade zeichnen

Um gerade Pfade zu zeichnen, legen Sie die Anfangs- und Endkoordinaten der Linie fest.

```
var canvas = document.getElementById("meinCanvas");  
var ctx = canvas.getContext("2d");  
① ctx.lineWidth = 4;  
② ctx.beginPath();  
ctx.moveTo(0,0);  
ctx.lineTo(200,100);  
ctx.stroke();
```

Gerader Pfad

- ① Definieren Sie mit `lineWidth` die Breite der Pfadlinie in Pixel.
- ② Rufen Sie die Methode `beginPath` auf, um einen neuen Pfad zu zeichnen.
- ③ Rufen Sie die Methode `moveTo` auf.
 - ✓ Das erste Argument der Methode `moveTo` legt die x-Koordinate des Ausgangspunkts der Linie fest.
 - ✓ Das zweite Argument ist die y-Koordinate des Ausgangspunkts der Linie.
- ④ Rufen Sie die Methode `lineTo` auf.
 - ✓ Das erste Argument der Methode `lineTo` ist die x-Koordinate des Endpunkts der Linie.
 - ✓ Das zweite Argument ist die y-Koordinate des Endpunkts der Linie.
- ⑤ Rufen Sie die Methode `stroke` auf. Diese zeichnet den Pfad.



Eine gerade Linie zeichnen

Beispiel: Pfade zu gefüllten Formen zusammensetzen (*kap10\canvas-raute.html*)

Mit der Funktion `fill` zeichnen Sie gefüllte Formen, indem Sie mehrere gerade Pfade zu einer Gesamtform zusammensetzen können, die Sie anschließend gefüllt darstellen können.

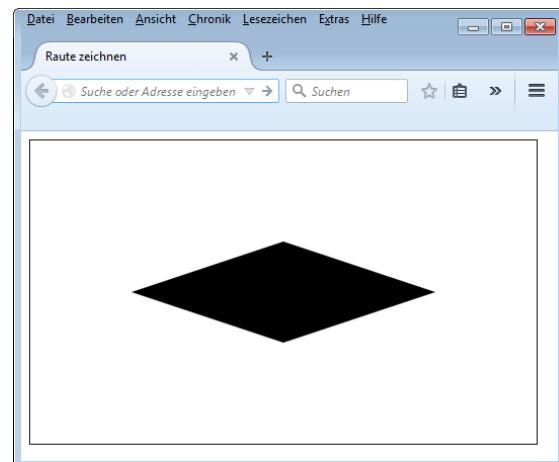


Das Prinzip der Funktion `fill` gleicht dem der Funktion `stroke`.

```
var canvas = document.getElementById("meinCanvas");
var ctx = canvas.getContext("2d");
ctx.beginPath();
① ctx.moveTo(100, 150);
② ctx.lineTo(250, 100);
③ ctx.lineTo(400, 150);
④ ctx.lineTo(250, 200);
⑤ ctx.fill();
```

Gefüllte Raute

- ① Setzen Sie den Anfangspunkt des Pfads (die linke Ecke der Raute) auf die Koordinaten x 100, y 150.
- ② Setzen Sie die obere Ecke der Raute auf die Koordinaten x 250, y 100.
- ③ Setzen Sie die rechte Ecke der Raute auf die Koordinaten x 400, y 150.
- ④ Setzen Sie die untere Ecke der Raute auf die Koordinaten x 250, y 200.
- ⑤ Zeichnen Sie eine gefüllte Raute.



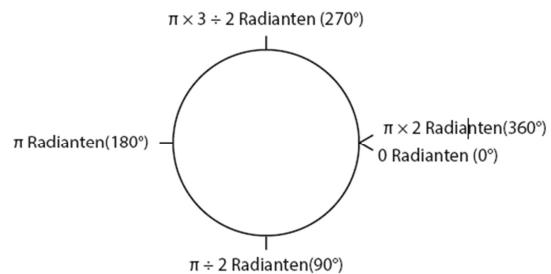
Eine gefüllte Raute zeichnen

Kreise und Bögen zeichnen

Kreise und Bögen lassen sich mit der Methode `arc` auf den Canvas zeichnen.

```
arc(x, y, r, Anfangswinkel, Endwinkel, Richtung)
```

- ✓ x und y sind die **Mittelpunktkoordinaten** des Kreises.
- ✓ r ist der **Radius** des Kreises.
- ✓ Anfangswinkel und Endwinkel werden in Radianen angegeben (siehe schematische Darstellung rechts). Sie bestimmen damit, welchen Teil des Kreises Sie zeichnen möchten, um einen **Bogen** zu erstellen.
- ✓ Richtung kann `true` oder `false` sein: Mit `true` wird der Bogen im Uhrzeigersinn gezeichnet, mit `false` im Gegenuhzeigersinn.



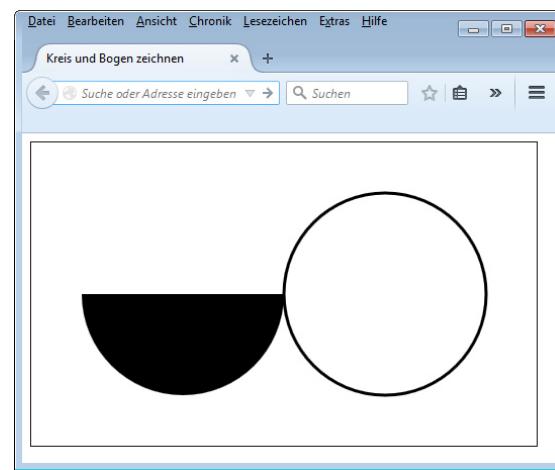
Beispiel: Kreis und Halbkreis zeichnen ([kap10/canvas-kreis-bogen.html](#))

Mit dem folgenden Code erzeugen Sie einen gefüllten Halbkreis mit einem Radius von 100 Pixeln und daneben einen vollen, un gefüllten Kreis mit demselben Radius.

```
var canvas = document.getElementById("meinCanvas");
var ctx = canvas.getContext("2d");
ctx.beginPath();
① ctx.arc(150, 150, 100, 0, Math.PI, false);
② ctx.fill();
③ ctx.lineWidth = 3;
④ ctx.beginPath();
⑤ ctx.arc(350, 150, 100, 0, Math.PI * 2, false);
ctx.stroke();
```

Voller Kreis und Halbkreis

- ① Setzen Sie den Mittelpunkt des Kreisbogens auf die Koordinaten x 150, y 150. Den Radius setzen Sie auf 100. Der Halbkreis beginnt bei 0 und reicht bis π (siehe schematische Darstellung oben). Mit `false` zeichnen Sie den Kreis im Gegenuhrzeigersinn
- ② Zeichnen Sie einen gefüllten Bogen.
- ③ Setzen Sie die Umrissstärke auf 3 px.
- ④ Legen Sie den Mittelpunkt des Kreises auf die Koordinaten x 350, y 150 fest, den Radius auf 100. $\pi \times 2$ ergibt einen vollen Kreis (siehe schematische Darstellung oben).
- ⑤ Zeichnen Sie einen Kreisumriss.



Einen gefüllten Bogen und einen Kreis zeichnen

10.5 Text auf den Canvas zeichnen

Neben geraden und gebogenen Formen und Linien können Sie auch Text auf dem Canvas darstellen. Wesentliche Eigenschaften und Methoden sind:

<code>font</code>	Definiert die Schrifteigenschaften
<code>textAlign</code>	Definiert die Textausrichtung. <ul style="list-style-type: none"> ✓ <code>left</code> oder <code>start</code> richten den Text links an der x-Koordinate aus. ✓ <code>center</code> zentriert den Text um die x-Koordinate. ✓ <code>right</code> oder <code>end</code> richten den Text rechts an der x-Koordinate aus.
<code>textBaseline</code>	Definiert die vertikale Ausrichtung des Textes. <ul style="list-style-type: none"> ✓ <code>alphabetic</code> (Standardeinstellung) verwendet die y-Koordinate als Textgrundlinie. ✓ <code>top</code> verwendet die y-Koordinate als Oberkante der Mittellänge, ✓ <code>middle</code> zentriert den Text vertikal.

<code>fillText(text, x, y)</code>	Zeichnet gefüllten Text	Hallo Welt!
<code>strokeText(text, x, y)</code>	Zeichnet ungefüllten Text	Hallo Welt!

Beispiel: Gefüllten Text auf dem Canvas zentrieren (*kap10\canvas-text-gefuellt.html*)

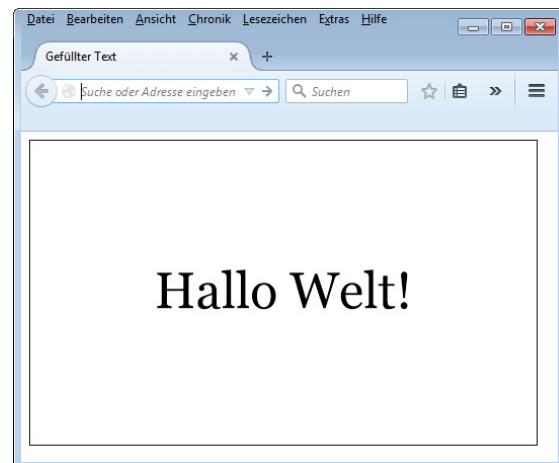
```
var canvas = document.getElementById("meinCanvas");
var ctx = canvas.getContext("2d");
① ctx.font = "50px Georgia";
② ctx.textAlign = "center";
③ ctx.textBaseline = "middle";
④ ctx.fillText("Hallo Welt!", meinCanvas.width/2, meinCanvas.height/2);
```

Gefüllten Text auf den Canvas zeichnen

- ① Die Schriftformatierung wird auf die Schriftart Georgia mit der Größe 50px festgelegt.
- ② Der Text wird zentriert ausgerichtet.
- ③ Der Text wird auf der Grundlinie zentriert.
- ④ Der gefüllte Text „Hallo Welt!“ wird auf den Canvas gezeichnet. Die Berechnungen `meinCanvas.width/2` und `meinCanvas.height/2` als Koordinatenangaben sorgen dafür, dass der Text horizontal und vertikal auf dem Canvas zentriert wird.



`strokeText` wenden Sie genauso an wie `fillText`.



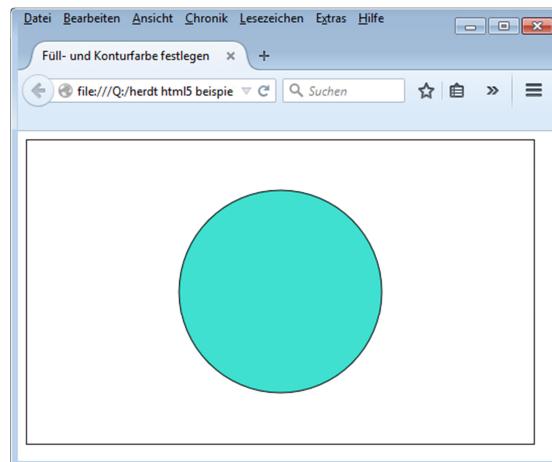
Text auf dem Canvas zentrieren

Beispiel: Die Füll- und Rahmenfarben eines Canvas-Elements festlegen (*kap10\canvas-farben*)

- ✓ Die Füllfarbe eines auf dem Canvas gezeichneten Elements ändern Sie mit der Eigenschaft `fillStyle` des Zeichenbereichs.
- ✓ Für die Konturfarbe gibt es die Eigenschaft `strokeStyle`.

```
var canvas = document.getElementById("meinCanvas");
var ctx = canvas.getContext("2d");
① ctx.fillStyle = "turquoise";
ctx.beginPath();
ctx.arc(250, 150, 100, 0, Math.PI * 2, false);
ctx.fill();
② ctx.strokeStyle = "black";
ctx.beginPath();
ctx.arc(250, 150, 100, 0, Math.PI * 2, false);
ctx.stroke();
```

- ① Am einfachsten ist es, der Eigenschaft `fillStyle` einen Farbnamen als String zu übergeben.
- ② Die Kontur des zweiten Kreises wird mit der Eigenschaft `strokeStyle` festgelegt



Füll- und Konturfarbe festlegen

Farbverläufe

Ebenso können die Kontur oder Fläche von Canvas-Elementen mit Farbverläufen gefüllt werden. Dabei können sowohl lineare als auch radiale Verläufe erzeugt werden.

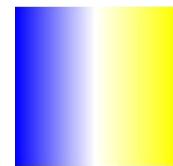
<code>createLinearGradient(x1,y1,x2,y2)</code>	Diese Funktion erzeugt einen linearen Farbverlauf. <ul style="list-style-type: none"> ✓ <code>x1</code> ist die X-Koordinate des Anfangspunkts des Verlaufs. ✓ <code>y1</code> ist die Y-Koordinate des Anfangspunkts. ✓ <code>x2</code> ist die X-Koordinate des Verlaufsendpunkts. ✓ <code>y2</code> ist die Y-Koordinate des Endpunkts.
<code>createRadialGradient(x1,y1,r1,x2,y2,r2)</code>	Diese Funktion erzeugt einen radialen Farbverlauf. Hier kommen gegenüber dem linearen Verlauf noch die Parameter <code>r1</code> und <code>r2</code> hinzu: <ul style="list-style-type: none"> ✓ <code>r1</code> ist der Radius des Anfangskreises. ✓ <code>r2</code> ist der Radius des Endkreises.

- ▶ Nachdem Sie den Verlauf erzeugt haben, fügen Sie zwei oder mehr Farben hinzu. Zum Hinzufügen der Farben und ihrer Positionen verwenden Sie die Methode `addColorStop()`. Die Farbpositionen können einen Wert zwischen 0 und 1 annehmen.
- ▶ Um den Verlauf anschließend zu nutzen, weisen Sie ihm die Eigenschaft `fillStyle` oder `strokeStyle` zu und zeichnen die Form.

Beispiel: Ein Quadrat mit einem linearen Farbverlauf füllen ([kap10\canvas-linearer-verlauf.html](#))

	<pre>var canvas = document.getElementById("meinCanvas"); var ctx = canvas.getContext("2d"); ① var verlauf=ctx.createLinearGradient(150,50,350,50); ② verlauf.addColorStop(0,"blue"); ③ verlauf.addColorStop(0.5,"white"); ④ verlauf.addColorStop(1,"yellow"); ⑤ ctx.fillStyle=verlauf; ⑥ ctx.fillRect(150,50,200,200);</pre>
--	--

- ① Der lineare Verlauf beginnt bei den Koordinaten $x = 150$, $y = 100$ und endet bei $x = 350$, $y = 50$. Er hat also eine Länge von 200 Pixel und verläuft horizontal.
- ② Der Verlauf beginnt bei Position 0 mit blau.
- ③ Bei der Hälfte des Verlaufs (0.5) wird eine weiße Farbmarke gesetzt.
- ④ Als Stil wird die Füllung festgelegt.
- ⑤ Der Verlauf wird in Quadratform mit den Abmessungen 200 x 200 Pixel gezeichnet.



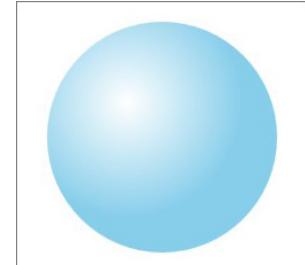
Linearen Verlauf erzeugen

Beispiel: Einen Kreis mit einem radialen Farbverlauf füllen ([kap10\canvas-radialer-verlauf.html](#))

Der folgende Code erzeugt einen radialen Farbverlauf, der gegenüber dem gezeichneten Kreis ein wenig nach links und oben verschoben ist. So entsteht der Eindruck einer dreidimensionalen Kugel.

```
var canvas = document.getElementById("meinCanvas");
var ctx = canvas.getContext("2d");
① var verlauf=ctx.createRadialGradient(220,120,0,220,120,100);
② verlauf.addColorStop(0,"white");
③ verlauf.addColorStop(1,"skyblue");
④ ctx.fillStyle = verlauf;
  ctx.beginPath();
⑤ ctx.arc(250,150,100,0,Math.PI * 2);
  ctx.fill();
```

- ① Der Anfang des radialen Verlaufs hat die Koordinaten x 220, y 120. Der Radius des inneren Farbkreises beträgt 0 px. Das Ende des radialen Verlaufs hat dieselben Koordinaten, der Radius beträgt jedoch 100 (die Endgröße des geplanten Kreises).
- ② Der Verlauf beginnt bei Position 0 mit weiß.
- ③ Am Ende des Verlaufs (1) geht der Verlauf in die Farbe skyblue über.
- ④ Als Stil wird die Füllung festgelegt.
- ⑤ Der Verlauf wird in voller Kreisform mit einem Radius von ebenfalls 100 Pixel gezeichnet. Die Koordinaten sind gegenüber dem Verlauf nach rechts unten verschoben, damit der gewünschte 3D-Effekt entsteht.



Kugel mit 3D-Effekt zeichnen

10.6 Übungen

Übung 1: Geografische Position kontinuierlich ermitteln

Übungsdatei: --

Ergebnisdatei: uebung1.html

1. Erzeugen Sie eine Webseite, die Ihre geografische Position kontinuierlich ermittelt.
2. Falls vorhanden, testen Sie die Seite mit einem iPhone oder anderen mobilen Gerät mit GPS-Funktion.

Übung 2: Dynamischen Kreismittelpunkt setzen

Übungsdatei: --

Ergebnisdatei: uebung2.html

1. Der Code in der Beispieldatei *radialer-verlauf-zentriert.html* ist so geschrieben, dass sich der Kreis mit dem radialen Verlauf in der Mitte des Canvas befindet.
2. Ändern Sie den Code so ab, dass der Kreis samt Verlauf dynamisch in der Canvasmitte bleibt, egal welche Größe der Canvas erhält. Schreiben Sie dabei einen möglichst schlanken Code.

11 Kurzeinstieg in AJAX

In diesem Kapitel erfahren Sie

- ✓ was sich hinter dem Begriff AJAX verbirgt
- ✓ wie Sie Daten anfordern, ohne die Webseite erneut laden zu müssen

Voraussetzungen

- ✓ Fundierte JavaScript-Kenntnisse

11.1 Was steckt hinter AJAX?

AJAX (**A**synchronous **J**ava**S**cript **a**nd **X**ML) und ist eine Möglichkeit, Informationen im Browser auszutauschen, ohne die Webseite neu laden zu müssen. AJAX basiert auf JavaScript und XML und ermöglicht das Aktualisieren von bestimmten Bereichen einer Webseite. Im Gegensatz zu klassischen Webanwendungen muss nicht die ganze Webseite neu geladen werden. So können interaktive Webseiten erstellt werden, die echten Applikationen ähneln. Beispielsweise können Suchergebnisse schon während der Eingabe eines Suchbegriffs angezeigt werden.

Das ist auch das bis heute am häufigsten verwendete und bekannteste Einsatzgebiet. Wenn Sie z. B. bei Suchmaschinen nach einem Begriff oder bei den meisten großen Versandhäusern im Internet nach einem bestimmten Produkt suchen wollen, werden Ihnen direkt bei Eingabe verschiedene Suchvorschläge unterbreitet.

The screenshot shows a search interface with the following elements:

- Search Bar:** Contains the text "bürostuhl".
- Clear Button:** A red button labeled "lös!".
- Search Results:**
 - Sortimentsvorschläge:** Includes "Bürostuhl" and "Bürostuhl Topstar".
 - Suchvorschläge:** Includes "Gernot Bürostuhl", "bürostuhl point 60", and "Bürostuhl Open Point Sy".
 - Product Listings:** Shows items like "Topstar Bürostuhl Autosynchron®...", "Topstar Bürostuhl X-Pander blau, ...", "Gernot Steifensand Bürostuhl Profi...", "Topstar Bürostuhl Tec 50 schwarz", and "SITWELL Bürostuhl Water blau". Each listing includes a small chair icon and a link to the product details.
- User Navigation:** Includes links for "Suche", "Markenübersicht", "Zubehörsuche", "Mein Konto", "anmelden", "Mein Shop", "Bestellschein", "Service", and "0 Produkte".

Eingabemasken mit den Vorschlägen zu dem eingegebenen Begriff

Mit Ajax ist es auch möglich, Internet-Anwendungen zu programmieren, wie zum Beispiel Google Text & Tabellen (docs.google.com). Mit dieser Ajax-Anwendung können Sie Text und Tabellen direkt im Browser bearbeiten, ohne ein Textbearbeitungs- bzw. Tabellenkalkulationsprogramm auf Ihrem System installiert zu haben. Ein weiteres Beispiel sind die Oberflächen der verschiedenen Webmailer. Mit AJAX können Sie mittlerweile die E-Mails wie in einem E-Mail-Programm verschieben und löschen.

Mit Ajax erstellte Komponenten stehen in Programmier-Bibliotheken (sogenannten Frameworks) zur Verfügung. Webseitenentwickler können auf die fertigen Komponenten (Widgets) zugreifen und sich das Schreiben des Programmcodes ersparen.

Das W3C hat mittlerweile eine einheitliche Definition für das XMLHttpRequest-Objekt zur Standardisierung vorgeschlagen, um eine vollständige Kompatibilität der XMLHttpRequest-Implementierungen zu erzielen.

11.2 Das XMLHttpRequest-Objekt

Das Objekt erstellen

Um die Funktionalität des Objekts nutzen zu können, müssen Sie es zuerst erzeugen.

```
<script>
  var ajaxhttp = new XMLHttpRequest();
</script>
```

Damit haben Sie das Objekt `ajaxhttp` erstellt und können nun auf die Eigenschaften und Methoden der neuen Browser zugreifen.

Da Microsoft dieses Objekt erst ab dem Internet Explorer 7 implementiert hat, sollten Sie gegebenenfalls zusätzlich auch das ActiveX-Objekt `Microsoft.XMLHTTP` instanzieren, falls der eine oder andere Nutzer noch diesen veralteten Browser verwendet. Dies geschieht relativ einfach über eine entsprechende Abfrage:



```
<script>
  var ajaxhttp;
  if (window.XMLHttpRequest) {
    ajaxhttp = new XMLHttpRequest();
  } else if (window.ActiveXObject) {
    ajaxhttp = new ActiveXObject("Microsoft.XMLHTTP");
  }
</script>
```

Wenn der Browser das Objekt `XMLHttpRequest` kennt, soll dieses verwendet werden, ansonsten wird auf das ActiveX-Objekt zurückgegriffen.

Eine HTTP-Anfrage erstellen

Nachdem Sie das Objekt erstellt haben, müssen Sie eine Verbindung zur Webseite aufbauen, die Daten zurückliefern soll. Diese Verbindung definieren Sie über die Methode `open()`.

<code>open(Methode, URL, [asynchron])</code>	Neben der URL der aufzurufenden Datei geben Sie als Methode, mit der die Daten übertragen werden sollen, POST oder GET an. Aus Sicherheitsgründen sollte der Methode POST der Vorzug gegeben werden. Wie die Kommunikation zwischen Browser und Webserver ablaufen soll, legen Sie mit einem Wert <code>true</code> oder <code>false</code> fest. Standardmäßig wird <code>asynchronous</code> (<code>true</code>) kommuniziert. Dies hat den Vorteil, dass der Browser nicht sofort auf die Antwort warten muss. Beim synchronen Datenaustausch (<code>false</code>) würde der Browser nicht mehr reagieren, wenn der Webserver zu langsam arbeitet oder wenn die Verbindung schlecht ist.
--	--

```
ajaxhttp.open("POST", "datei.php", true);
ajaxhttp.open("GET", "datei.php", false);
```

Da bei einer asynchronen Kommunikation nicht auf die Antwort gewartet wird, gibt es die Eigenschaft `onreadystatechange`. In dieser definieren Sie eine beliebige Funktion, die auch Callback-Funktion genannt wird.

```
/* Variante 1 */
ajaxhttp.onreadystatechange = function() { }

/* oder Variante 2 */
ajaxhttp.onreadystatechange = Funktion;

function Funktion {
}
```

Die hinterlegte Funktion wird immer dann aufgerufen, wenn sich der Zustand der HTTP-Anfrage ändert. Folgende Zustände gibt es:

Zustand	Beschreibung
0	Das Objekt ist nicht initialisiert.
1	Das Objekt baut eine Verbindung auf.
2	Die Anfrage wurde erfolgreich gesendet.
3	Das Objekt wartet auf die Rückantwort.
4	Das Ergebnis wurde zurückgeliefert.

Eine HTTP-Anfrage durchläuft hierbei immer alle Zustände von 0 bis 4. Im Normalfall müssen Sie nur den Zustand 4 beachten, da hier die zurückgelieferten Daten über verschiedene Eigenschaften abgefragt werden können.

<code>responseText</code>	Der zurückgelieferte Wert wird als JavaScript-Zeichenkette bereitgestellt.
<code>responseXML</code>	Der Wert wird als XML-Element zurückgeliefert.
<code>status</code>	Hier können Sie den HTTP-Statuscode abfragen.
<code>statusText</code>	Dies liefert den Beschreibungstext des Statuscodes.

Die HTTP-Anfrage senden

Mit der Methode `send()` übergeben Sie die definierte Anfrage an den Webserver und starten damit die Anforderung der Daten.

```
ajaxhttp.send(null);
```

Beispiel: Inhalt einer Textdatei anzeigen (`kap11\ajax-get.html`)

In diesem Beispiel soll eine Webseite erstellt werden, die im Hintergrund eine einfache Textdatei lädt und deren Inhalt über die Eigenschaft `innerHTML` an einer bestimmten Stelle innerhalb der Webseite einfügt. Außerdem soll der Status ausgegeben werden, den der Server nach der Anfrage zurücksendet.



Aufgrund der Sicherheitseinstellungen des Internet Explorers kann dieses Beispiel mit den Standardeinstellungen des Internet Explorer nicht ausgeführt werden. Sie setzen die Sicherheitseinstellungen über das Zahnradsymbol in der rechten oberen Programmfensterecke, den Befehl *Internetoptionen* und die Registerkarte *Sicherheit* herab.

```
<html>
<head>
<meta charset="utf-8">
<title>Text nachträglich laden</title>
<script>
①   var ajaxhttp;
②   if (window.XMLHttpRequest) {
      ajaxhttp = new XMLHttpRequest();
    } else if (window.ActiveXObject) {
      ajaxhttp = new ActiveXObject("Microsoft.XMLHTTP");
    }

③   if (ajaxhttp != null) {
④     ajaxhttp.open("GET", "text.txt", true);
⑤     ajaxhttp.onreadystatechange = Textausgabe;
⑥     ajaxhttp.send(null);
    }

⑦   function Textausgabe() {
⑧     if (ajaxhttp.readyState == 4) {
⑨       document.getElementById("textstelle").innerHTML = ajaxhttp.responseText;
       document.getElementById("status").innerHTML = 'Status: ' +
                                                 ajaxhttp.status + ' ('+ ajaxhttp.statusText + ')';
     }
   }
</script>
</head>
<body>
  <h2>AJAX - Text laden</h2>
  <p>Der nachfolgende Text wird dynamisch nach dem Laden der Webseite geladen
     und angezeigt.</p>
⑩  <div id="textstelle">Dieser Text wird ausgetauscht...</div>
  <div id="status">Statusausgabe</div>
</body>
</html>
```

- ① Zu Beginn definieren Sie die Variable ajaxhttp.
- ② Im nächsten Schritt überprüfen Sie, ob der Browser das aktuelle XMLHttpRequest-Objekt versteht. Entsprechend wird das Objekt in der Variablen ajaxhttp instanziert.
- ③ War das Zuweisen des Objekts erfolgreich, also != null, so wird in diese Schleife verzweigt, in der die Eigenschaften festgelegt werden.
- ④ Über die GET-Methode soll die Datei *text.txt*, die einen beliebigen Inhalt haben kann, angefordert werden. Es soll eine asynchrone Verbindung aufgebaut und somit nicht direkt auf die Antwort gewartet werden.
- ⑤ Bei Änderungen des zurückgelieferten Status soll in die Funktion Textausgabe verzweigt werden.
- ⑥ Die Anforderung wird abgesendet.
- ⑦ In der Funktion wird als Erstes überprüft, ob ein Ergebnis zurückgeliefert wurde (*readyState == 4*).
- ⑧ Ist dies der Fall, wird in der Webseite das Element mit der ID *textstelle* ermittelt. Über die Eigenschaft *innerHTML* können Sie beliebigen Text übergeben, der dann vom Browser angezeigt wird. In diesem Fall ist es der Inhalt der Rückgabe (*responseText*).
- ⑨ Am Element *status* werden der Status sowie die entsprechende Erklärung ausgegeben. Beachten Sie, dass der Status nur von einem Server zurückgeliefert wird und nicht lokal auf dem Rechner funktioniert. Sie müssen also die entsprechenden Dateien auf einen Webserver laden und von dort abrufen.
- ⑩ Die beiden div-Container mit den IDs *textstelle* und *status* werden festgelegt.

Möchten Sie Daten über die Methode POST anfordern, müssen Sie zusätzlich die Eigenschaft `setRequestHeader` definieren. Die erforderliche Angabe sieht folgendermaßen aus.

```
ajaxhttp.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
```



Die Methode GET wird verwendet, wenn Sie nur Text übertragen wollen. Die Daten werden dabei als Werte von Variablen durch ein Fragezeichen an eine URL angehängt. Die Menge der Daten ist beschränkt und vom jeweils verwendeten Browser abhängig.



Die übermittelten Daten sind dabei in der Adresszeile des Browsers sichtbar.

Mit der Methode POST können Sie auch binäre Daten in beliebiger Menge übertragen. Dazu werden die Daten codiert und im Nachrichtenrumpf übertragen. Die Daten sind für den Anwender nicht im Browser sichtbar.

Beispiel: Daten über POST anfordern (*kap11\ajax-post.html*)

Bei einer Anforderung der Daten über die Methode POST geschieht die Zuweisung der Eigenschaften folgendermaßen:

```
/* ... */
if (ajaxhttp != null) {
    ajaxhttp.open("POST", "gettext.php", true);
    ajaxhttp.onreadystatechange = Textausgabe;
    ajaxhttp.setRequestHeader("Content-Type",
                             "application/x-www-form-urlencoded");
    ajaxhttp.send(null);
}
/* ... */
```

Wie Sie in diesem Beispiel sehen können, ist auch das Aufrufen von Skript-Dateien möglich. In diesem Fall ist es eine PHP-Datei, es kann aber auch jede andere Skriptsprache sein, wie z. B. ASP.NET, cgi, Ruby, Python.

Übergabe von Parametern

Auch die Übergabe von zusätzlichen Parametern, die dann entsprechend ausgewertet werden können, an die Skript-Dateien ist möglich. Je nach verwendeter Methode GET oder POST sind die Angaben unterschiedlich. Bei der Methode GET brauchen Sie die Parameter einfach nur an den angeforderten Dateinamen anzuhängen.

```
ajaxhttp.open("GET", "gettext.php?param1=wert1&param2=wert2", true);
```

Bei der Methode POST können die Parameter nicht an den Dateinamen angehängt werden. Hier übergeben Sie die Parameter mit der Methode `send()`.

```
ajaxhttp.open("POST", "gettext.php", true);
/* ... */
ajaxhttp.send("param1=wert1&param2=wert2");
```

Die HTTP-Anfrage abbrechen

Mit den bisherigen Methoden haben Sie keine Möglichkeit, darauf zu reagieren, wenn keine Rückmeldung vom Server kommt. Das heißt, dass der Browser im schlimmsten Fall ewig auf eine Rückmeldung wartet und der Anwender mit Sicherheit mehrfach die Anfrage ausprobieren wird.

Für diesen Fall ist es zweckmäßig, dass Sie mit der Methode `setTimeout()` auf eine Funktion verweisen, die nach dem Ablauf einer festgelegten Zeitspanne die Anfrage beendet. Die hierfür notwendige Methode des Objekts XMLHttpRequest lautet `abort()`:

```
ajaxhttp.abort();
```

Beispiel: Rückantwort prüfen und abbrechen (*kap11\ajax-abort.html*)

Das bisherige Beispiel wird erweitert, indem eine zeitliche Überprüfung der Rückantwort erfolgt und entsprechend abgebrochen wird.

```
<script>
    var ajaxhttp;
    var warten;

    if (window.XMLHttpRequest) {
        ajaxhttp = new XMLHttpRequest();
    } else if (window.ActiveXObject) {
        ajaxhttp = new ActiveXObject("Microsoft.XMLHTTP");
    }

    if (ajaxhttp != null) {
        ajaxhttp.open("GET", "zufallszeit.php", true);
        ajaxhttp.onreadystatechange = Textausgabe;
        ajaxhttp.send(null);
    }
}

function Abbruch() {
    ajaxhttp.abort();
    document.getElementById("textstelle").innerHTML = '<span
        style="color:red;">Zeitüberschreitung: Es wurde keine Antwort empfangen.
        Die Verbindung wurde abgebrochen...</span>';
}

function Textausgabe() {
    if (ajaxhttp.readyState == 4) {
        document.getElementById("textstelle").innerHTML = ajaxhttp.responseText;
        document.getElementById("status").innerHTML = 'Status: ' +
            ajaxhttp.status + ' (' + ajaxhttp.statusText + ')';
    }
}
</script>
```

- ① Die Variable `warten` wird global definiert, um darauf in allen Funktionen zugreifen zu können.
- ② In der Schleife, in der die Abfrage definiert wird, legen Sie mit der Methode `setTimeout()` fest, dass nach 4 Sekunden die Funktion `Abbruch()` aufgerufen werden soll. Die Zeitangabe erfolgt hierbei in Millisekunden.

- ③ In dieser Funktion geben Sie an, dass die Anfrage mittels `abort()` abgebrochen werden soll. Am Element `textstelle` soll ein entsprechender Text in roter Farbe ausgegeben werden.
- ④ In der bisherigen Funktion `Textausgabe` geben Sie zusätzlich die `window`-Methode `clearTimeout()` an. Damit wird nach einer erfolgreichen Datenübertragung der Warteprozess mit dem Funktionsaufruf `Abbruch()` abgebrochen. Würde dies nicht erfolgen, würde immer nach 4 Sekunden in die Abbruchsfunktion verzweigt werden, egal ob die Übertragung erfolgreich war oder nicht.

Beispiel: Zufallszahl erzeugen (`kap11\zufallszeit.php`)

```
<?php  
$zufall = rand(1, 10);  
sleep($zufall);  
  
$lines = implode('', file('text.txt'));  
echo $lines;  
?>
```

Dieses PHP-Skript erzeugt eine Zufallszahl zwischen 1 und 10 und führt über den PHP-Befehl `sleep()` eine entsprechende Verzögerung aus. Erst nach dieser Zeit wird der Inhalt der Datei `text.txt` geladen und ausgegeben.

AJAX - Text laden

Der nachfolgende Text wird dynamisch nach dem Laden der Webseite geladen und angezeigt.

Text, der per AJAX über eine HTTP-Anfrage in diese Webseite nachträglich eingefügt wurde.

Status: 200 (OK)

Nachträglich geladener Inhalt

AJAX - Text laden

Der nachfolgende Text wird dynamisch nach dem Laden der Webseite geladen und angezeigt.

Zeitüberschreitung: Es wurde keine Antwort empfangen. Die Verbindung wurde abgebrochen...

Status: 0 ()

Der Empfang der Antwort dauerte zu lange

11.3 Was ist noch möglich?

Wie Sie sehen, ist AJAX eigentlich ein weiteres JavaScript-Objekt, das es ermöglicht, Daten anzufordern und zu empfangen. Mithilfe der Methode `innerHTML()` lassen sich alle Elemente einer Webseite überschreiben, ohne dass die Webseite neu geladen werden muss.

Dabei sind Sie nicht nur auf die Anzeige von Text beschränkt. Mit AJAX und einer entsprechenden Skriptsprache sind Sie in der Lage, auch Grafiken auszutauschen, um beispielsweise eine interaktive Bildergalerie zu erzeugen. In dem Fall muss das Skript auf dem Server nur den entsprechenden Quellcode-Text für die Grafik zurückliefern.

Auch die auf dem Webserver verwendete Skriptsprache, die das Ergebnis der AJAX-Anfrage zurückliefert, ist nicht vorgegeben. Hier können Sie z. B. PHP, ASP.Net, CGI, Python oder Ruby verwenden. Wichtig ist, dass die Skriptsprache auf die Anfrage reagieren und ein entsprechendes Ergebnis zurückliefern kann.

A1 Cascading-Style-Sheet-Referenz



Dieser Abschnitt stellt keine vollständige Referenz dar, sondern bietet einen Ausschnitt der Möglichkeiten. Eine ausführliche Referenz finden Sie beispielsweise unter <http://www.css4you.de/>.

Schrift

Eigenschaft	Werte	Elemente	Vererbung
font-family	Namen der Schriften	alle	✓
font-style	normal (Standard), italic, oblique	alle	✓
font-variant	normal (Standard), small-caps	alle	✓
font-weight	normal (Standard), bold, bolder, lighter, 100, 200, 300, 400, 500, 600, 700, 800, 900	alle	✓
font-size	xx-small, x-small, small, medium, large, x-large, xx-large, larger, smaller Größe, Prozentangabe	alle	✓
font	Kurzform: style variant weight size/ line-height family	alle	✓

Farben und Hintergründe

Eigenschaft	Werte	Elemente	Vererbung
color	Farbangaben als Schlüsselwort, hexadezimaler, dezimaler oder Prozentwert	alle	✓
background-color	Farbangaben als Schlüsselwort, hexadezimaler, dezimaler oder Prozentwert	alle	✓
background-image	Adresse der Grafik	alle	
background-repeat	repeat, repeat-x, repeat-y, no-repeat	alle	
background-attachment	scroll, fixed	alle	
background-position	Mischung von top, bottom, left, center, right	innerhalb eines Blocks	
background	Kurzform: image color repeat position attachment	alle	

Texteigenschaften

Eigenschaft	Werte	Elemente	Vererbung
word-spacing letter-spacing	normal <i>Länge</i>	alle	✓
white-space	normal (<i>Standard</i>), pre, nowrap	alle	✓
text-decoration	none (<i>Standard</i>), underline, overline, line-through, blink	alle außer 	
vertical-align	baseline (<i>Standard</i>), sub, super, top, text-top, middle, bottom, text-bottom <i>Prozentangabe</i>	innerhalb eines Blocks	
text-transform	none (<i>Standard</i>), capitalize, uppercase, lowercase	alle	✓
text-align	left (<i>Standard</i>), right, center, justify	innerhalb eines Blocks	✓
text-indent	<i>Länge, Prozentangabe (negative Werte sind erlaubt)</i>	innerhalb eines Blocks	✓
line-height	normal <i>Größe, Prozentangabe</i>	alle	✓

Listen

Eigenschaft	Werte	Elemente	Vererbung
list-style-type	disc, circle, square, decimal, lower-roman, upper-roman, lower-alpha, upper-alpha	Listen-elemente	✓
list-style-image	none <i>Adresse der Grafik</i>	Listen-elemente	✓
list-style-position	outside (<i>Standard</i>), inside	Listen-elemente	✓
list-style	<i>Kurzform:</i> type position image	Listen-elemente	✓

Box-Eigenschaften

Eigenschaft	Werte	Elemente	Vererbung
margin-top margin-right margin-bottom margin-left	auto <i>Abstand, Prozentangabe</i>	alle	
margin	<i>Kurzform: top right bottom left</i>	alle	
padding-top padding-right padding-bottom padding-left	<i>Abstand, Prozentangabe</i>	alle	
padding	<i>Kurzform: top right bottom left</i>	alle	
border-top-width border-right-width border-bottom-width border-left-width	thin, medium, thick <i>Größe</i>	alle	
border-width	<i>Kurzform: top right bottom left</i>	alle	
boder-style	none, dotted, dashed, solid, double, groove, ridge, inset, outset	alle	
border-color	<i>Farbangaben (eine Angabe für alle Ränder oder vier Angaben für jeden einzelnen Rand)</i>	alle	
border-top border-right border-bottom border-left	<i>Kurzform: width style color</i>	alle	
border	<i>Kurzform: width style color</i>	alle	

Positionieren

Eigenschaft	Werte	Elemente	Vererbung
position	static (<i>Standard</i>), relative, absolute, fixed, inherit	alle	
float	none (<i>Standard</i>), left, right, inherit	alle	
clear	none, both, left, right	alle	
top left right bottom	auto (<i>Standard</i>), inherit <i>Größe, Prozentangabe</i>	alle	
width height	auto <i>Breite, Prozentangabe</i>	alle	
z-index	auto, inherit <i>Zahl</i>	alle	
overflow	visible (<i>Standard</i>), hidden, scroll, auto, inherit	innerhalb eines Blocks	
clip	auto (<i>Standard</i>), rect(top, right, bottom, left)	alle Elemente mit absoluter Positionierung	
display	none, block, inline, list-item	alle	
visibility	inherit (<i>Standard</i>), visible, hidden, collapse	alle	

A2 JavaScript-Objektreferenz

Globale Objekte

Objekt	Eigenschaft	Methoden
Array	length	concat, join, pop, push, reverse, shift, slice, sort, splice, unshift
Date	-	getDate, getDay, getFullYear, getHours, getMilliseconds, getMinutes, getMonth, getSeconds, getTime, getTimezoneOffset, getUTCDate, getUTCFullYear, getUTCHours, getUTCMilliseconds, getUTCMinutes, getUTCMonth, getUTCSeconds, getYear, parse, setDate, setFullYear, setHours, setMilliseconds, setMinutes, setMonth, setSeconds, setTime, setUTCDate, setUTCFullYear, setUTCHours, setUTCMilliseconds, setUTCMinutes, setUTCMonth, setUTCSeconds, setYear, toGMTString, toLocaleString, UTC
Math	Konstanten: E, LN10, LN2, LOG10E, LOG2E, PI, SQRT1_2, SQRT2	abs, acos, asin, atan, atan2, ceil, cos, exp, floor, log, max, min, pow, random, round, sin, sqrt, tan
Number	MAX_VALUE, MIN_VALUE, NaN, NEGATIVE_INFINITY, POSITIVE_INFINITY	toExponential,toFixed, toPrecision, toSource, toString, valueOf
RegExp	global, ignoreCase, input, lastIndex, source	compile, exec, test
String	length	anchor, big, blink, bold, charAt, charCodeAt, concat, fixed, fontcolor, fontsize, fromCharCode, indexOf, italics, lastIndexOf, link, match, replace, search, slice, small, split, strike, sub, substr, subString, sup, toLowerCase, toUpperCase

Fenster: window-Objekt

Eigenschaft	Methoden	Ereignisabfrage
closed, defaultStatus, innerHeight, innerWidth, length, locationbar, menubar, name, opener, outerHeight, outerWidth, pageXOffset, pageYOffset, parent, personalbar, scrollbars, self, status, statusbar, toolbar, top,	alert, back, blur, captureEvents, clearInterval, clearTimeout, close, confirm, disableExternalCapture, enableExternalCapture, find, focus, forward, handleEvent, home, moveBy, moveTo, open, print, prompt, releaseEvents, resizeBy, resizeTo, routeEvent, scroll, scrollBy, scrollTo, setInterval, setTimeout, setResizable, stop	onBlur, onError, onFocus, onLoad, onMove, onResize, onUnload

Unterobjekte des **window**-Objekts

Objekt	Eigenschaft	Methoden
document	(siehe document-Objekt)	
frames	(siehe frames-Objekt)	
history	current, length, next, previous	back, forward, go
location	hash, host, hostname, href, pathname, port, protocol, search	reload, replace

Frames: **frames**-Objekt

Eigenschaft	Methoden	Ereignisabfrage
length, name, parent, self, window	siehe Methoden des window-Objekts	siehe Ereignisse des window-Objekts

Browserinformation: **navigator**-Objekt

Eigenschaft	Methoden	Ereignisabfrage
appCodeName, appName, appVersion, mimeTypes, platform, plugins, userAgent	javaEnabled	-

Bildschirm: **screen**-Objekt

Eigenschaft	Methoden	Ereignisabfrage
availHeight, availLeft, availTop, availWidth, colorDepth, height, pixelDepth, width	-	-

Dokument: **document**-Objekt

Eigenschaft	Methoden	Ereignisabfrage
alinkColor, bgColor, cookie, domain, embeds, fgColor, height, ids, lastModified, linkColor, referrer, title, URL, vlinkColor, width	close, getSelection, open, write, writeln	onClick, onDoubleClick, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseUp

Unterobjekte des `document`-Objekts

Objekt	Eigenschaft	Methoden	Ereignisabfrage
anchors	length, name, text, x, y	-	-
applets	length	-	-
forms	action, elements, encoding, length, method, name, target	reset, submit	onReset, onSubmit
images	border, complete, height, hspace, length, lowsrc, name, src, vspace, width	-	onAbort, onError, onKeyDown, onKeyPress, onKeyUp, onLoad
links	hash, host, hostname, href, length, pathname, port, protocol, search, target, text, x, y	-	onClick, onDoubleClick, onMouseDown, onMouseOut, onMouseUp, onMouseOver

Formular: `forms`-Objekt

Objekt	Eigenschaft	Methoden	Ereignisabfrage
button	form, name, type, value	blur, click, focus	onBlur, onClick, onFocus, onMouseDown, onMouseUp
checkbox	checked, defaultChecked, form, name, type, value	blur, click, focus	onBlur, onClick, onFocus
elements	checked, defaultChecked, defaultValue, form, name, type, value	blur, click, focus, select	onBlur, onClick, onFocus, onSelect
hidden	form, name, type, value	-	-
option	defaultSelected, index, length, selected, text, value	-	-
password	defaultValue, form, name, type, value	blur, focus, select	onBlur, onFocus
radio	checked, defaultChecked, form, name, type, value	blur, click, focus	onBlur, onClick, onFocus
reset	form, name, type, value	blur, click, focus	onBlur, onClick, onFocus
select	form, length, name, options, selectedIndex, type	blur, focus	onBlur, onChange, onFocus
submit	form, name, type, value	blur, click, focus	onBlur, onClick, onFocus

Objekt	Eigenschaft	Methoden	Ereignisabfrage
text	defaultValue, form, name, type, value	blur, focus, select	onBlur, onChange, onFocus, onKeyDown, onKeyPress, onKeyUp, onSelect
textarea	defaultValue, form, name, type, value	blur, focus, select	onBlur, onChange, onFocus, onKeyDown, onKeyPress, onKeyUp, onSelect

Reservierte Wörter

Diese Liste enthält die bereits in Gebrauch befindlichen reservierten Wörter sowie die für den zukünftigen Sprachausbau geplanten reservierten Wörter.

Auch die für den zukünftigen Sprachausbau geplanten reservierten Wörter dürfen nicht für Funktions- oder Variablennamen verwendet werden.



abstract	delete	function	null	throw
boolean	do	goto	package	throws
break	double	if	private	transient
byte	else	implements	protected	true
case	enum	import	public	try
catch	export	in	return	typeof
char	extends	instanceof	short	var
class	false	int	static	void
const	final	interface	super	volatile
continue	finally	long	switch	while
debugger	float	native	synchronized	with
default	for	new	this	

A3 DOM-Referenz

Das Document Object Model (DOM)

Das Objektmodell, das von den Browsern der neuen Generation genutzt wird, besteht grundsätzlich aus drei Methoden. Mit denen können Sie auf jeden Elementknoten einer Webseite zugreifen.

Methode	Beschreibung
getElementById	Damit können Sie auf alle Elemente zugreifen, die eine eindeutige Bezeichnung <code>id</code> besitzen, z. B. <code><div id="rahmen"></code> .
getElementsByName	Hiermit selektieren Sie Elemente, die nicht unbedingt einen eindeutigen Namen haben, z. B. <code><input type="radio" name="auswahl"></code> .
getElementsByTagName	Sämtliche HTML-Tags innerhalb einer Webseite erhalten Sie mit dieser Methode, z. B. <code>getElementsByTagName ("td")</code> , um alle Tabellenzellen ansprechen zu können.

@		Canvas, Formen zeichnen Canvas, Kreise zeichnen Canvas, Linien zeichnen Canvas, Text zeichnen Canvas, zeichnen	146 148 147 149 145	E
@media	44	Canvas-2D-Kontext Canvas-2D-Kontext, API canvas-Element case Case-sensitive CGI	145 141 145 68 62 58	Ebene Ebene anzeigen und verbergen Ebene positionieren Eigenschaft Eingabe überprüfen Eingabefeld Eingabefenster Element, unsichtbares Elementausschnitt
<		checkbox clip:rect() Codecs concat confirm() Container Container, Breite und Höhe zuweisen	112 40 51 104 83 126 27	Encoder Endlosschleife Ereignisabfrage Ereignisse escape() Escape-Sequenzen eval()
A		Container, Seitenlayout aufbauen continue	31 71	Fallauswahl, mehrseitige false Favicon Feed Feedreader Felder Felder, versteckte figCaption figure float footer for forms Formulardaten Frameworks function Funktion Funktion, Wertübergabe an Funktionen referenzieren Funktionsaufruf
active	9	Date()	100	
hover	9	Datentypen	62, 64	
link	9	Datentypkonvertierung	66	
visited	9	Datum	100	
abort	159	Datumsangabe	104	
AJAX	154	default	68	
anchors	109	defaultChecked	112	
Anfügeoperator	65	DefaultValue	111	
API	140	Dekomprimierung	51	
API, Definition	140	Dekrement	64	
applets	110	Destop first	48	
Application Cache, API	140	Dezimalzahl	62	
Application Programming Interface	140	DHTML	124	
appName	87	Dialogfenster	81	
appVersion	87	div	25	
Array()	103	div-Container	24	
article	25	div-Container, formatieren	27	
ASCII-Code	92	div-Container, für das Seitenlayout		
aside	25	erstellen	25	
audio	52	div-Container, positionieren	29	
Audio einbinden	53	div-Container, schweben lassen	28	
Ausschnitt eines Elements	40	div-Container, verschachteln	33	
Auswahlliste	112	Document Object Model	124	
B		document.all	125	
Barrierefreiheit	7	document.documentElement	125	
Bedingungsoperator	66	document.layers	125	
Bedingungsprüfung	70	Doorway-Seiten	21	
Bestätigungsfenster	83	do-while	70	
Bilder, Imagemaps	17	Drag & Drop, API	140	
blur()	111	Dynamic HTML	124	
Boolesche Werte	66			
break	68, 70			
Browsereigenschaft	87			
Browserweiche	125			
C				
Callback-Funktion	155			
Canvas, Bögen zeichnen	148			
canvas, Farbverläufe zeichnen	151			
D				G
Date()	100	Ganzzahlen	62	
Datentypen	62, 64	Geografische Position	142	
Datentypkonvertierung	66	Geolocation, API	141	
Datum	100	Geolocation-API	141	
Datumsangabe	104	getCurrentPosition()	142	
default	68	Gleitkomazahlen	63	
defaultChecked	112	Google-Maps-Karte	143	
DefaultValue	111	GPS	141	
Dekomprimierung	51			
Dekrement	64			
Destop first	48			
Dezimalzahl	62			
DHTML	124			
Dialogfenster	81			
div	25			
div-Container	24			
div-Container, formatieren	27			
div-Container, für das Seitenlayout				
erstellen	25			
div-Container, positionieren	29			
div-Container, schweben lassen	28			
div-Container, verschachteln	33			
Document Object Model	124			
document.all	125			
document.documentElement	125			
document.layers	125			
Doorway-Seiten	21			
do-while	70			
Drag & Drop, API	140			
Dynamic HTML	124			

H

header	25
height	28
hidden	112
history	95
Hotspots	16
HTTP-Anfrage	155
Hyperlinks, Imagemaps	16

I

if	67
if-else	67
Image()	118
Imagemaps	16
Imagemaps erstellen	16
Imagemaps erzeugen	17
images	118
Inkrement	64
Interpreter	66
isNaN()	80

J

Java-Applets	110
JavaScript	58
JavaScript testen	60
JavaScript, Objekte	86
JavaScript, Sonderzeichen	63
JavaScript, Standardobjekte	97
Javascript:	79
JavaScript-APIs	140
join	104

K

Kompression	51
Komprimierung	51
Konkatenationsoperator	65
Kontrollfelder	112
Kontrollstrukturen	67

L

Layout zentrieren	35
link	19
link rel=	19, 21
links	109
Local Storage, API	140
location	96
Logische Verlinkung	18

M

main	25
Math	99
max-width	44
Methode	86
MIME	20, 54, 56
min-width	44
Mobile first	48
MouseOver-Effekt	118
MP3	52
MPEG-4	52
Multimedia	50
Multimediaformate	51

N

nav	25
Navigationselemente	6
Navigationselemente benennen	7
Navigationselemente gestalten	7
Navigationselemente, Einheitlichkeit	7
Navigationselemente, platzieren	6
Navigationsleiste	6
Navigationsleiste aus Text-Hyperlinks	8
Navigationsleiste mit CSS gestalten	9
Navigationsleiste mit Mouseover-Effekten	9
Navigationsleiste, Buttons	10
Navigationsleiste, horizontale	8
Navigationsmenü	6, 12
Navigationsmenü mit CSS anordnen	13
Navigationsmenü vertikal ausrichten	15
Navigationsmenü, HTML-Struktur	12
navigator	87
navigator.geolocation	141
Netscape Navigator	125
Newsfeed	18
Notationsregeln, break/continue	70
number()	80

O

Objekt Array	103
Objekt Date	100
Objekt document	90
Objekt event	91
Objekt history	95
Objekt location	96
Objekt Math	99
Objekt navigator	87
Objekt String	97
Objekt window	89

Objekte

Objekte, Eigenschaften	86
------------------------	----

Objekte, eingegebene	86
----------------------	----

Objekte, Konzept	86
------------------	----

Objekte, Methoden	86
-------------------	----

Objekt-Eigenschaft	87
--------------------	----

Objekt-Methode	87
----------------	----

OGG	52
-----	----

on(Ereignis)	76
--------------	----

onClick	75
---------	----

onreadystatechange	156
--------------------	-----

open	155
------	-----

Operator	64
----------	----

Operator, ternärer	66
--------------------	----

Operatoren, Datentypen	66
------------------------	----

options []	113
------------	-----

Optionsfeld	112
-------------	-----

Optionsfeld abfragen	116
----------------------	-----

overflow	38
----------	----

P

parseFloat()	80
parseInt()	80
password	111
Perl	58
PHP	58
Plug-ins	50
Plug-ins einbinden	50
pop()	104
position	29
Positionierung, absolute	30
Positionierung, relative	31
Positionsobjekt	142
Postfixoperator	64
push	104

R

Radio Button	112
Random()	99
Reset	114
responseText	156
reverse()	104

S

Schaltfläche	114
Schleife, beenden	70
Schleifen	68
script	59
section	26

select()	111	V	
Selektion	68	Var	61
Semantische Tags	25	Variablen	61
send	156	Variablen, Operatoren	64
Server Sent Events, API	140	Vergleichsoperator	65
setRequestHeader	158	Verlinkung, logische	18
setTimeout()	102	Verweis	109
SetTimeout()	89	Verweisanker	109
shift()	104	Verzögerungszeit	89, 102
Sicherheitsabfrage beim Zurücksetzen	117	video	54
Sichtbarkeit	39, 126	Videoformate	52
slice()	104	Videos einbinden	54
Sonderzeichen	63	Videos in HTML5 einbinden	51
sort()	104	viewport	45
Sound einbinden	52	visibility	39, 126
Sound in HTML5 einbinden	51		
Soundkonsole	53		
splice()	104	W	
Standort des Benutzers	143	Web Workers, API	140
Standort eines Nutzers	141	Webauftritt, Navigation	6
Standortermittlung	143	Webdesign, responsives	43
status	156	while	69
Steuerzeichen	63	width	28
String	63	Window.Alert()	89
String()	80, 97	window-Objekt	92
Submit	114	write()	91
switch	68	writeln()	91
T		X	
Tabellen, Nachteile	24	XMLHttpRequest	155
template	26		
Ternär	66		
Text	111		
textarea	111	Z	
Texteditor	58	Zahlen	62
Trennung von Struktur und Darstellung	8	Zählschleife	69
true	64	Zeichenkette	63, 97
Tunnel-Seiten	21	Zeit	100
typeof	66	z-index	42
		Zufallsgenerator	99
		Zufallszahl	99
		Zurücksetzen, Sicherheitsabfrage	117
U			
Überlauf	38		
Uhrzeit	100		
Umleitung auf eine andere Adresse	21		
unescape()	80		
Unterprogramm	74		

Impressum

Matchcode: HTML5F

Autorin: Isolde Kommer

Fachlektorat: Katja Wengler

Redaktion: Andrea Weikert

Produziert im HERDT-Digitaldruck

2. Ausgabe, September 2015

HERDT-Verlag für Bildungsmedien GmbH
Am Kümmerling 21-25
55294 Bodenheim
Internet: www.herdt.com
E-Mail: info@herdt.com

© HERDT-Verlag für Bildungsmedien GmbH, Bodenheim

Alle Rechte vorbehalten. Kein Teil des Werkes darf in irgendeiner Form (Druck, Fotokopie, Mikrofilm oder einem anderen Verfahren) ohne schriftliche Genehmigung des Verlags reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

Dieses Buch wurde mit großer Sorgfalt erstellt und geprüft. Trotzdem können Fehler nicht vollkommen ausgeschlossen werden. Verlag, Herausgeber und Autoren können für fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung übernehmen.

Wenn nicht explizit an anderer Stelle des Werkes aufgeführt, liegen die Copyrights an allen Screenshots beim HERDT-Verlag. Sollte es trotz intensiver Recherche nicht gelungen sein, alle weiteren Rechteinhaber der verwendeten Quellen und Abbildungen zu finden, bitten wir um kurze Nachricht an die Redaktion.

Die in diesem Buch und in den abgebildeten bzw. zum Download angebotenen Dateien genannten Personen und Organisationen, Adress- und Telekommunikationsangaben, Bankverbindungen etc. sind frei erfunden. Eventuelle Übereinstimmungen oder Ähnlichkeiten sind unbeabsichtigt und rein zufällig.

Die Bildungsmedien des HERDT-Verlags enthalten Verweise auf Webseiten Dritter. Diese Webseiten unterliegen der Haftung der jeweiligen Betreiber, wir haben keinerlei Einfluss auf die Gestaltung und die Inhalte dieser Webseiten. Bei der Bucherstellung haben wir die fremden Inhalte daraufhin überprüft, ob etwaige Rechtsverstöße bestehen. Zu diesem Zeitpunkt waren keine Rechtsverstöße ersichtlich. Wir werden bei Kenntnis von Rechtsverstößen jedoch umgehend die entsprechenden Internetadressen aus dem Buch entfernen.

Die in den Bildungsmedien des HERDT-Verlags vorhandenen Internetadressen, Screenshots, Bezeichnungen bzw. Beschreibungen und Funktionen waren zum Zeitpunkt der Erstellung der jeweiligen Produkte aktuell und gültig. Sollten Sie die Webseiten nicht mehr unter den angegebenen Adressen finden, sind diese eventuell inzwischen komplett aus dem Internet genommen worden oder unter einer neuen Adresse zu finden. Sollten im vorliegenden Produkt vorhandene Screenshots, Bezeichnungen bzw. Beschreibungen und Funktionen nicht mehr der beschriebenen Software entsprechen, hat der Hersteller der jeweiligen Software nach Drucklegung Änderungen vorgenommen oder vorhandene Funktionen geändert oder entfernt.

Hochschulversion