Eingebettete Frames

Stand: 04.07.2022

In HTML eingebettete Frames

Eingebettete Frames (auch als **Inline-Frames** bezeichnet) ermöglichen es, an einer gewünschten Stelle innerhalb des sichtbaren Inhalts eines HTML-Dokuments Inhalte aus anderen Dateien oder Quellen einzubinden. Dazu stellt HTML das `iframe`-Element
zur Verfügung (iframe = inline frame). Wie der Name schon vermuten lässt, hat dieses Element Inline-Charakter, d.h., es kann – ähnlich wie Grafikreferenzen – auch mitten im Text notiert werden.

> Zum reinen Einbetten einer anderen Quelle kann übrigens auch das

object-Element verwendet werden, das im Gegensatz zum iframe-Element zum strict-Standard von HTML gehört. Bei Verwendung des iframe-Elements müssen Sie dagegen den Dokumenttyp für die HTML-Variante transitionalangeben. Erforderlich ist die Verwendung des iframe-Elements jedoch dann, wenn Hyperlinks ihr Verweisziel im Fenster des Inline-Frames öffnen sollen. Genau diesen Anwendungsfall werden wir im nachfolgenden Beispiel behandeln. Der Quelltext
der Datei mit dem eingebetteten Frame lautet:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html lang="de">
<head>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1">
<title>Eingebetteter Frame</title>
</head>
<body style="text-align:center">
<h1>Blindtext</h1>
<p>
<a href="seite1.htm" target="content">Seite 1</a> |
<a href="seite2.htm" target="content">Seite 2</a> |
<a href="seite3.htm" target="content">Seite 3</a> |
<a href="seite4.htm" target="content">Seite 4</a>
</p>
<p>
<iframe name="content" src="seite1.htm"
frameborder="0"
style="width:60%;
height:200px;
```

```
border:thin solid #C0C0C0">
Ihr Browser unterstützt leider keine eingebetteten Frames!
</iframe>
</p>
</body>
</html>
```

Info Das `iframe`-Element wird mit Start- und End-Tag notiert. Zwischen `<iframe>` und `</iframe>` kann Inhalt für Browser notiert werden, die das `iframe`-Element nicht kennen oder interpretieren.
Dabei dürfen auch beliebige Block- und Inline-Elemente notiert werden. Wie beim `frame`-Element wird auch beim `iframe`-Element eine Default-Quelle über das `src`-Attribut referenziert. Es kann sich um einen beliebigen
URI handeln, also um eine lokal referenzierte Datei oder auch um eine absolute Internetadresse. Wie auch beim `frame`-Element ist das `name`-Attribut wichtig, wenn Linkziele im eingebetteten Frame geöffnet werden sollen.

Weitere Angaben zum Frame-Fenster wie Breite und Höhe

des Frame-Fensters werden im obigen Beispiel zeitgemäß über **CSS** gelöst. Die Angabe `width:60%` im Beispiel bedeutet: 60% der Breite des Elternelements. Dieses ist das umgebende `p`-Element, welches sich im normalen
Textfluss innerhalb des `body`-Elements befindet. Da es ein Block-Element ist, nimmt es die maximal verfügbare Breite ein, also die gesamte Breite des Anzeigefensters abzüglich der Default-Ränder, die der Browser setzt. Mithilfe von CSS lässt
sich auch der Rahmen des eingebetteten Frames optimal gestalten. Im Beispiel haben wir einen dünnen grauen Rahmen (`border:thin solid #C0C0C0`) gewählt. Dennoch ist zusätzlich das HTML-Attribut `frameborder="0"` notiert. Damit wird
explizit der Frame-eigene Rahmen unterdrückt, den der Internet Explorer andernfalls unabhängig von der CSS-Rahmendefinition anzeigt. Die Links im Beispiel haben im einleitenden `<a>`-Tag ein Attribut `target="content"`. Dadurch
wird der Fensterbezug zum `iframe`-Element hergestellt, das mit `name="content"` definiert wird. Die Verweisziele werden im Frame-Fenster geöffnet. Scrollleisten zeigt der Browser nach Bedarf an. Durch `scrolling="no"`können Sie die Anzeige von Scrollleisten verhindern.

Fixe Bereiche ohne Frames

Wie bereits erwähnt, werden Frames in den meisten Fällen nur eingesetzt, um einen fixen Bereich zu haben, der nicht mit scrollt (auch als **Non-Scrolling-Region** bezeichnet). In diesem Artikel möchten wir zeigen, wie so etwas auch ohne Frames
funktioniert - aus der Überlegung heraus, dass es einfach zu schade ist, die zahlreichen Nachteile von Frames in Kauf zu nehmen, wenn das gewünschte Ziel auch anders erreichbar ist. Das HTML-Dokument hat folgenden Inhalt:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html lang="de">
<head>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1">
<title>Fixer Bereich</title>
<link rel="stylesheet" type="text/css" href="fix.css">
</head>
<body>
<div id="navigation">
<a href="s2.html">Seite 2</a>
<br>
<a href="s3.html">Seite 3</a>
<br>
<a href="s4.html">Seite 4</a>
</div>
<div id="content">
<h1>Seite 1</h1>
<p>viel Inhalt</p>
...
<p>viel Inhalt</p>
</div>
</body>
</html>
```

Info Die beiden Bereiche erhalten die `id`-Namen `navigation` und `content`. In der eingebundenen CSS-Datei fix.css werden folgende Formate definiert:

```
#
navigation {
    position: fixed;
    top: 20 px;
    left: 20 px;
    padding - right: 20 px;
    border - right: blue solid 4 px;
}#
content {
    margin - left: 200 px;
    margin - right: 20 px;
}
```

Info Der Bereich `navigation` wird mit der Angabe `position:fixed` fest positioniert. Beim Scrollen wandert sein Inhalt nicht mit. Die restlichen Definitionen betreffen Randabstände sowie einen blauen dicken Rahmen rechts

von den Links. Der Bereich `content` soll im normalen Textfluss bleiben, erhält also keine Angabe zur Positionierung. Mit `margin-left:200px` wird ein hinreichend großer linker Rand geschaffen, der die Anzeige der Navigationslinks
in diesem Bereich erlaubt.

Eigentlich ist das bereits alles und in Browsern, welche `position:fixed` interpretieren, funktioniert auch alles wie gewünscht.

> Der Inhalt kann gescrollt werden, während die Links fest an ihrer Position bleiben.

Leider interpretiert der MS Internet Explorer bis einschließlich Version 6.0 `position:fixed` nicht. Stattdessen ignoriert er die Positionierungsangabe, was durchaus ein korrektes Verhalten ist, aber dazu führt, dass die beiden Bereiche beide
im normalen Textfluss verbleiben. Der `div`-Bereich für die Navigationslinks verhält sich also wie ein gewöhnliches Blockelement im Textfluss. Es steht an der Stelle, wo es notiert wird (zuerst), und nimmt die gesamte verfügbare Breite ein,
so dass der `div`-Bereich für den Inhalt erst unterhalb davon angezeigt wird. Dieses Verhalten ist jedoch nicht erwünscht. Andererseits kann ein so verbreiteter Browser nicht einfach ignoriert werden. Wir benötigen also eine Behelfslösung.
Dank anderer, proprietärer Fähigkeiten des Internet Explorers ist das Verhalten von `position:fixed` "simulierbar".

Workaround für den Internet Explorer

Dazu wird zunächst der Inhalt der CSS-Datei **fix.css** wie folgt geändert:

```
body > #navigation {
    position: fixed;
    top: 20 px;
    left: 20 px;
    padding - right: 20 px;
    border - right: blue solid 4 px;
}
body > #content {
    margin - left: 200 px;
    margin - right: 20 px;
}
```

Info Geändert wird die Syntax der **Selektoren** dahingehend, dass von den CSS-2.0-spezifischen Adressierungsmöglichkeiten für Elemente Gebrauch gemacht wird. Diese werden vom Internet Explorer

ebenfalls nicht interpretiert. Als Ergebnis
ignoriert dieser Browser die Formatdefinitionen für beide `div`-Bereiche komplett. Im zweiten Schritt bekommt der Internet Explorer stattdessen eine nur für ihn lesbare CSS-Datei spendiert. Dazu wird im HTML-Dokument - am besten hinter dem
`<link>`-Tag, welches das normale Stylesheet einbindet - Folgendes notiert:

```
<!--[if gte IE 5]>
<link rel="stylesheet" type="text/css" href="ie-fix.css">
<![endif]-->
```

Info Der gesamte Code stellt aus HTML-Sicht einen Kommentar dar. Eigentlich sollten Browser den Inhalt von Kommentaren ignorieren und die meisten tun das auch. Doch nicht so der Internet Explorer. Hier kann der Kommentar interpretierbaren Inhalt
haben. Das Konstrukt `[if gte IE 5]>....` `<![endif]` wird, wenn ohne Leerraum nach bzw. vor den Kommentarklammern notiert, als Verarbeitungsanweisung interpretiert. `[if gte IE 5]>` bedeutet so viel wie: "Wenn
ein Internet Explorer mit Versionsnummer größer oder gleich 5.0 das hier liest, dann soll er den HTML-Code bis zu `<![endif]` trotz Kommentar interpretieren." Genutzt wird diese Fähigkeit in unserem Fall, um mittels `<link>`-Tag
eine weitere CSS-Datei speziell für den Internet Explorer einzubinden. Im Beispiel nennen wir sie ie-fix.css. Diese spezielle CSS-Datei erhält folgenden Inhalt:

```
html, body {
    overflow: hidden;
    width: 100 % ;
    height: 100 % ;
}#
navigation {
    position: absolute;
    top: 20 px;
    left: 20 px;
    padding - right: 20 px;
    border - right: blue solid 4 px;
}#
content {
    position: absolute;
    top: 0 px;
    left: 200 px;
    padding - top: 20 px;
    height: expression(document.body.clientHeight - 20 + "px");
    width: expression(document.body.clientWidth - 200 + "px");
    overflow: auto;
}
```

Info Zunächst wird für die Basiselemente `html` und `body` festgelegt, dass diese die volle Breite und Höhe einnehmen. Gleichzeitig wird bestimmt, dass Inhalte, die länger oder breiter sind, einfach abgeschnitten werden (`overflow:hidden`).
Die beiden `div`-Bereiche mit den `id`-Namen `navigation` und `content` werden daraufhin beide absolut positioniert. Bis auf die Art der Positionierung sind die Formatdefinitionen für den Bereich `navigation`die gleichen wie in der normalen CSS-Datei. Bei den Definitionen für `content` gibt es hingegen einige Unterschiede. Mit `top` und `left` muss dessen gewünschte Anfangsposition bestimmt werden, da er ja absolut positioniert
wird. Eigentlich soll er 20 Pixel von oben beginnen. Damit der Internet Explorer die vertikale Scrollleiste jedoch wie üblich oben im Fenster beginnen lässt, weisen wir `top:0px` zu und schaffen den gewünschten Abstand nach oben über den Umweg
`padding-top:20px`.

Besonders spannend sind die Zuweisungen an die CSS-Eigenschaften

`height` und `width`. Anstelle eines festen Werts wird ein Wert zugewiesen, der mithilfe von JScript errechnet wird. Der Internet Explorer erlaubt das Zuweisen gewisser Funktionen an CSS-Eigenschaften, unter anderem die Funktion
`expression()`. Diese ermöglicht das Ausführen von JavaScript/JScript-Code. In den beiden Beispielangaben werden die Breite und die Höhe des zu positionierenden Bereichs aus der tatsächlichen Breite und Höhe bestimmt, die der in **HTML**in diesem Bereich notierte Inhalt einnimmt. Die beiden Objekteigenschaften `document.body.clientWidth` und `document.body.clientHeight` liefern die Werte für den gesamten `body`-Bereich. Abgezogen werden die gewünschten
Startpositionen des Inhaltsbereichs (der Obenwert bei der Höhe und der Linkswert bei der Breite). Ausdrücklich zugewiesen wird dann noch `overflow:auto`. Somit wird der Bereich gescrollt, falls sein Inhalt es erfordert. In unserem Beispiel
gibt es nun keinerlei Unterschiede in der Funktionalität etwa zwischen Firefox, Opera und anderen einerseits und Internet Explorer andererseits. In allen Browsern bleiben die Navigationslinks fix an ihrer Position. Wenn Sie den hier beschriebenen Workaround
für fix positionierte Bereiche an anderer Stelle, beispielsweise oben, nutzen wollen, müssen Sie gegebenenfalls in der CSS-Datei für den Internet Explorer bei `html` und `body` nicht `overflow:hidden` angeben, sondern
`overflow-y:hidden`.

HTML mit Framesets und Frames

Das "Denken in Frames" erfordert zunächst einmal, sich Gedanken über mehrere unterschiedliche Dateien zu machen. Erforderlich sind eine Datei, in der das **Frameset** definiert wird, also die Aufteilung der Einzelfenster sowie je eine Datei
bzw. Quelle, die beim Aufruf der Frameset-Datei in die einzelnen Frame-Fenster geladen werden soll.

Datei mit Frameset-Definitionen

Das nachfolgende Listing zeigt den kompletten Quelltext einer Datei mit einer Frameset-Definition:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
"http://www.w3.org/TR/html4/frameset.dtd">
<html>
<head>
<title>Mein erstes Frameset</title>
</head>
<frameset cols="250,*">
<frame src="verweise.htm" name="Navigation">
<frame src="startseite.htm" name="Daten">
<noframes>
Ihr Browser kann diese Seite leider nicht anzeigen!
</noframes>
</frameset>
</html>
```

Info Zunächst fällt auf, dass eine andere Dokumenttyp-Deklaration verwendet wird. In Dateien, die ein Frameset definieren, müssen Sie diese Deklaration verwenden. Damit erfüllt eine solche Seite den HTML 4.01-Standard in der Variante für Framesets.

> Die weitere Struktur weicht ebenfalls stark von sonst üblichen HTML-Dokumenten ab.

Das html-Element enthält nicht wie sonst üblich die Elemente head und body als Inhalt, sondern die Elemente head und frameset. Das `body`-Element wird also durch das `frameset`-Element ersetzt. Im `head`-Element können jedoch die gleichen Daten stehen wie bei gewöhnlichen HTML-Dokumenten. Das `title`-Element ist Pflicht und andere Angaben, wie etwa `meta`- oder `link`-Elemente, sind ohne Ausnahme
erlaubt und meist auch sehr sinnvoll. Das `frameset`-Element (`<frameset>...</frameset>`) bestimmt die Verteilung der Fenster. Dazu kann es die beiden Attribute `rows=` und `cols=` enthalten.
Durch das `rows`-Attribut werden Fenster untereinander angeordnet (**rows** = Reihen), durch das `cols`-Attribut nebeneinander (cols = **columns** = Spalten). Im obigen Beispiel-Listing werden durch die Angabe `cols="250,*"` zwei Frame-Fenster nebeneinander angeordnet. Das linke Frame-Fenster soll 250 Pixel breit sein. Das rechte soll so breit sein, wie der Rest des Browser-Anzeigefensters es erlaubt. Die Wertzuweisung besteht also in kommagetrennten
Werten, wobei eine Zahl eine Pixelangabe bedeutet. Ferner sind Prozentangaben erlaubt wie etwa `cols="30%,*"`. Das Sternchen ist ein Platzhalter und bedeutet "die Breite soll vom Browser bestimmt werden".

> Selbstverständlich ist eine Aufteilung in mehr als zwei Spalten oder Reihen möglich.

So definiert beispielsweise `rows="200,250,*"` drei Frame-Fenster untereinander, wobei das erste 200 Pixel hoch ist, das zweite 250 Pixel hoch und das dritte soll so hoch sein wie der Rest der Anzeigehöhe des Browser-Fensters es ermöglicht.
Möglich sind auch Angaben wie `cols="100,3*,5*"`. Dies ist sinnvoll, wenn Sie ein oder zwei Frames mit einer Pixelangabe zu Höhe und Breite versehen möchten, die übrigen Höhen bzw. Breiten dagegen flexibel gestalten. Die Angabe im Beispiel
bedeutet: drei Frame-Fenster nebeneinander, wobei das erste 100 Pixel breit sein soll. Von der verbleibenden Breite soll das zweite Fenster drei Achtel einnehmen und das dritte fünf Achtel. Zahlen, gefolgt von einem Sternchen, werden als Verhältnisangabe
betrachtet, ähnlich wie Prozentangaben, jedoch bezogen auf den unbekannt großen, verbleibenden Teil des Browser-Fensters. Die Summe der Zahlen ist der gemeinsame Nenner. Die entstehenden Bruchwerte sind für den Browser die Vorgabewerte zur Berechnung
der tatsächlichen Framefenster-Breite bzw. -Höhe. Die Attribute `cols=` und `rows=` dürfen auch beide gleichzeitig verwendet werden. Ein Beispiel:

```
<frameset rows="100,*" cols="25%,*">
<frame src="obenlinks.html" name="f1">
<frame src="obenrechts.html" name="f2">
<frame src="untenlinks.html" name="f3">
<frame src="untenrechts.html" name="f4">
</frameset>
```

Info Durch eine solche Aufteilung wird ein Gitter mit so vielen Spalten mal Reihen wie angegeben erzeugt. Häufig werden jedoch auch andere Aufteilungen gewünscht, wie etwa "oben ein Streifen über die gesamte Breite und unten zwei Spalten". Für
solche Aufteilungen besteht die Möglichkeit, **Frameset-Definitionen** zu verschachteln:

```
<frameset rows="100,*">
<frame src="oben.html" name="f1">
<frameset rows="250,*">
<frame src="unenlinks.html" name="f2">
<frame src="untenrechts.html" name="f3">
</frameset>
</frameset>
```

Info Es werden also ein äußeres und ein inneres Frameset definiert. Das äußere Frameset besorgt mit `rows=` die Aufteilung in zwei Frame-Fenster untereinander. Das `frame`-Element für die obere Quelle wird auch erst einmal
notiert. Anstelle des `frame`-Elements für das untere Frame-Fenster wird jedoch ein weiteres, inneres `frameset`-Element notiert. Dieses bewirkt eine Aufteilung in zwei weitere, nebeneinander anzuordnende Frame-Fenster (`cols=`).
Für jeden kommaseparierten Wert, der bei einem `rows`- oder `cols`-Attribut zugewiesen wird, muss innerhalb des `frameset`-Elements also entweder ein `frame`-Element notiert werden, welches den

Fensterinhalt
angibt, oder ein weiteres, inneres `frameset`-Element, welches den Raum des Frame-Fensters für eine Aufsplittung in weitere Frame-Fenster nutzt. Während das `frameset`-Element aus Anfangs- und End-Tag besteht, ist `<frame...>`ein Standalone-Tag. In XHTML muss es folglich in der Form `<frameset ... />` notiert werden. Das `frame`-Element hat zwei wichtige Standardattribute. Beim `src`-Attribut wird der URI des Inhalts angegeben, der im
entsprechenden Frame-Fenster angezeigt werden soll. Es kann sich um eine lokal referenzierte Datei handeln oder auch um eine entfernte, absolute Internetadresse. Inhalte können HTML-Dateien sein, aber natürlich auch andere Quellen, wie Grafiken, Flash-Movies,
PDF-Dateien oder Ähnliches. Auch dynamische Quellen wie PHP-Scripts können selbstverständlich in jedem einzelnen Frame-Fenster als Quelle angegeben werden. Bei Grafik- oder Multimedia-Quellen ist allerdings zu beachten, dass hierbei keine Möglichkeit
besteht, einen Alternativtext anzugeben. Das andere wichtige Attribut für jedes `frame`-Element ist das name-Attribut. Der frei vergebbare Name ist vor allem wichtig, wenn Seiten, die innerhalb eines **Frame-Fenstersangezeigt** werden, mit einem Link den Inhalt eines anderen Frame-Fensters verändern wollen. Bei solchen Links muss das andere Frame-Fenster unter dem hier vergebenen Namen mit adressiert werden.

Noframes-Bereiche

Zwischen `<frameset>` und `</frameset>` darf neben inneren `frameset`-Elementen und `frame`-Elementen noch ein weiteres Element stehen, ausgezeichnet durch `<noframes>`... `</noframes>`. Dieses Element erlaubt es, für Browser oder andere Client-Programme, welche keine Frames anzeigen können, einen alternativen Inhalt anzubieten. Ein Beispiel:

```
<frameset rows="100,*">
<frame src="oben.html" name="f1">
<frameset rows="250,*">
<frame src="unenlinks.html" name="f2">
<frame src="untenrechts.html" name="f3">
</frameset>
<noframes>
<h1>Liebe Besucher</h1>
<p>
Diese Site verwendet Frames.
Bitte rufen Sie die <a href="noframes.html"> Navigation für nicht-frame-fähige
Browser</a> auf!
</p>
</noframes>
</frameset>
```

Info Das `noframes`-Element darf am Ende einer `frameset`-Struktur vor dem schließenden `</frameset>`-Tag des äußersten Framesets notiert werden. Als Inhalt sind beliebige Block- und Inline-Elemente erlaubt.

Interessant ist, dass das `noframes`-Element auch in HTML-Dokumenten notiert werden darf, die innerhalb eines Framesets in einem Frame-Fenster angezeigt werden. Dort muss allerdings im Dokumenttyp die Transitional-Variante von **HTML**angegeben werden. Das `noframes`-Element kann dann als ganz normales Element im Dateikörper notiert werden. Innerhalb davon können z.B. Rückverweise auf eine Seite mit Navigationslinks notiert werden. Browser wie Lynx, die zwar Framesets erkennen, jedoch nicht anzeigen können, stellen den Inhalt von `noframes`-Elementen im Dateikörper dar. Browser, die Frames anzeigen können, ignorieren die Inhalte dagegen.

Frames erstellen

Mit Frames gibt es nun endlich die Möglichkeit, die Navigation mit den Inhaltsseiten zusammenzubringen. Dazu müssen Sie etwas umdenken. Bis jetzt galt immer: Der Browser lädt (ein!) HTML-Dokument, das wird angezeigt, fertig. Bei **Frames** ändert sich das ein wenig: Der Browser lädt ein HTML-Dokument, das so genannte Frameset. Dieses Dokument enthält keine Inhalte, sondern Verweise auf mehrere einzelne Unterframes. Diese einzelnen Unterframes sind genau die Dokumente, die wir auf den letzten Seiten zusammengestellt haben.

Dafür sind zwei neue Tags notwendig:

- `<frameset>` enthält eine Definition eines Framesets - also, dass der Browserbereich in mehrere Frames unterteilt werden soll.
- `<frame>` gibt einen einzelnen Frame an.

Dabei ändert sich auch die Dokumenttypangabe am Anfang des Dokuments, in dem das `<frameset>`-Element steht; hier müssen wir Folgendes verwenden:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

Info Bei den einzelnen Frame-Dokumenten dagegen ändert sich in Bezug auf den Dokumenttyp nichts. Erinnern Sie sich: Das sind herkömmliche Dokumente, die Sie auch einzeln und unabhängig voneinander im Browser laden könnten. Die Erstellung eines Framesets geht in wenigen Schritten vonstatten. Zu einigen dieser Schritte gibt es allerdings noch eine ganze Reihe an Hintergrundinformationen, die Sie im Folgenden jeweils direkt erwähnt finden.

- 1. Erstellen Sie ein Frameset mit dem `<frameset>`-Element.

```
<frameset>
</frameset>
```

- 2. Geben Sie im `cols`-Attribut die Breite der Frames an.

Es gibt mehrere Möglichkeiten, die Breite anzugeben. Am einfachsten ist es, Pixelwerte zu verwenden. So steht folgende Angabe für zwei Frames, der erste ist 200 Pixel, der zweite 650 Pixel breit:

```
<frameset cols="200,650">
</frameset>
```

Allerdings ist das nicht ganz optimal, denn das setzt die Gesamtbreite beider **Frames** auf 800 Pixel fest. Hat der Benutzer eine größere Auflösung, wird der Browser den fehlenden Platz irgendwie verwenden, was zu hässlichen Ergebnissen
führen kann. Aus diesem Grund können Sie den Browser anweisen, anstelle einer festen Pixelzahl die restliche zur Verfügung stehende Breite zu verwenden. Das geht mit dem Sternchen. Folgende Angabe macht den linken Frame 200 Pixel breit, der rechte Frame
nimmt die restliche Breite im **Browser** in Anspruch:

```
<frameset cols="200,*">
</frameset>
```

Dritte Möglichkeit: Sie verwenden prozentuale Werte. Wenn also der rechte Frame dreimal so breit sein soll wie der linke Frame, können Sie den Wert des `cols`-Attributs wie folgt setzen:

```
<frameset cols="25%,75%">
</frameset>
```

- 3. Fügen Sie pro Frame ein `<frame>`-Element innerhalb des `<frameset>`-Elements ein:

```
<frameset cols="200,*">
<frame />
<frame />
</frameset>
```

- 4. Geben Sie die Adresse des entsprechenden Frames im `src`-Attribut des `<frame>`-Tags an.

Folgende Angabe lädt in den linken Frame die Navigation und in den rechten Frame die Startseite:

```
<frameset cols="200,*">
<frame src="navigation.html" />
<frame src="start.html" />
</frameset>
```

- 5. Geben Sie jedem Frame noch einen eindeutigen Namen - im name-Attribut des <frame>-Tags.

```
<frameset cols="200,*">
<frame src="navigation.html" name="Navi" />
<frame src="start.html" name="Inhalt" />
</frameset>
```

Am Ende haben Sie folgende HTML-Datei:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
<!-- frameset.html -->
<html>
<head>
<title>Frames</title>
</head>
<frameset cols="200,*">
<frame src="navigation.html" name="Navi" />
<frame src="start.html" name="Inhalt" />
</frameset>
</html>
```

Sie sehen, dass das **HTML-Dokument** keinen <body>-Bereich mehr enthält, denn dieser wird in der Frameset-Datei nicht benötigt. Trotzdem ist die Ausgabe im Webbrowser wie gewünscht: zwei Frames, der eine 200 Pixel breit,
der andere die restliche Breite. Im linken Frame wurde das Navigationsdokument geladen (navigation.html), im rechten Frame die Startseite (start.html). Im Übrigen ist es auch möglich, Frames untereinander anzubringen. Dazu sind die beiden folgenden Modifikationen
vonnöten:

- Verwenden Sie das Attribut rows statt cols.
- Der Wert des Attributs rows bezeichnet dann natürlich die Höhe der einzelnen Frames und nicht mehr die Breite.

Nachfolgend noch einmal das Frameset von vorher, allerdings sind die Frames dann untereinander angeordnet:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
<!-- frameset-untereinander.html -->
<html>
<head>
```

```
<title>Frames</title>
</head>
<frameset rows="200,*">
<frame src="navigation.html" name="Navi" />
<frame src="start.html" name="Inhalt" />
</frameset>
</html>
```

Das Ergebnis sieht zugegebenermaßen etwas ungewohnt aus, aber es funktioniert: Die beiden Frames erscheinen ab sofort untereinander.

Frames für Seitenlayouts

Normalerweise zeigen `frame`-fähige Browser zwischen den **Frame-Fenstern** systemabhängig aussehende Rahmen an, die es dem Anwender unter anderem erlauben, das Fensterverhältnis durch Ziehen der Rahmen mit der Maus zu ändern.
Ein häufiger Wunsch von Seitenanbietern sind "unsichtbare Fensterrahmen". Dadurch erscheint dem Anwender die gesamte Seite wie aus einem Dokument, obwohl in Wirklichkeit ein Frameset am Werk ist.

> Meist wurde dieses Feature in der Vergangenheit deshalb verlangt,

um zwar eine nahtlose Ansicht im Browser-Fenster anzubieten, jedoch mit einem fixen Bereich für Logo, Navigation usw., der nicht mit scrollt. Der Wunsch ist nachvollziehbar, doch das Mittel, hierfür zu einem Frameset zu greifen, ist mittlerweile nicht
mehr zeitgemäß. Fixe Bereiche sollten mithilfe von CSS definiert werden. Generell wird die Rahmendicke wie im folgenden Beispiel bestimmt:

```
<frame src="index.htm" name="inhalt" frameborder="10">
```

Info Das `frameborder`-Attribut ist laut **HTML-Spezifikation** in `<frame>`-Tags erlaubt. Dort notiert, bestimmt es die Rahmendicke in Pixel, und zwar für alle Rahmen zwischen diesem Frame-Fenster und benachbarten
Frame-Fenstern. Um keine Rahmen anzuzeigen, wird der zugewiesene Wert auf 0 gesetzt. Notieren Sie die `frameborder`-Angabe sicherheitshalber in allen `<frame>`-Tags. Andernfalls erreichen Sie in den Browsern möglicherweise
nicht den gewünschten Effekt. Ein anderer Fall ist, dass Sie die Rahmen zwischen den Frame-Fenstern zwar optisch für sinnvoll halten, aber nicht möchten, dass der Anwender die Größe der Frame-Fenster damit verändern kann. Für diesen Fall gibt es ein weiteres
Attribut, wie das nachfolgende Beispiel zeigt:

```
<frame src="index.htm" name="inhalt" noresize>
```

Info Das Standalone-Attribut `noresize` (also "kein Resize", keine Größenänderung) verhindert für die Rahmen um das betroffene Frame-Fenster, dass diese mit der Maus in ihrer Position veränderbar sind. Beachten Sie, dass Sie bei XHTML `noresize="noresize"`notieren müssen.

Schließlich können Sie noch das Scrollverhalten von Frame-Fenstern beeinflussen.

Dazu können Sie im `<frame>`-Tag eines Fensters das Attribut `scrolling=` wahlweise mit den Werten `yes`, `no` oder `auto` versorgen: `yes` erzwingt Scrollleisten sogar dann, wenn der Inhalt gar kein Scrollen erfordert; `no` verhindert Scrollleisten in jedem Fall, auch dann, wenn der Inhalt Scrollen erfordert, und `auto`, die Voreinstellung, bewirkt, dass Scrollbars dann angezeigt werden, wenn es der Inhalt erfordert. Auf weitere Eigenschaften gehen wir an dieser Stelle nicht mehr ein, da diese entweder nicht zum **HTML-Standard** gehören oder nicht mehr zeitgemäß sind.

Frames verschachteln

So weit, so gut. Sie können jetzt also mehrere nebeneinander liegende Frames erstellen und auch mehrere untereinander liegende. Beides zusammen ist ein wenig schwieriger zu verstehen, aber dann auch sehr simpel umzusetzen.

Erinnern wir uns: Mit einem `<frameset>`-Element können Sie den Browserbereich in mehrere Frames unterteilen, entweder nebeneinander (`cols`) oder untereinander (`rows`). Wenn jetzt aber der Bereich sowohl horizontal als auch vertikal geteilt werden muss, müssen Sie beide Methoden kombinieren. Und das geht so: Zeichnen Sie sich am besten auf, wie die Frame-Struktur aussehen soll. Gehen wir zum Beispiel von vorhin zurück: links Navigation, rechts Inhalt. Angenommen, Sie möchten zur teilweisen Refinanzierung Ihres Webangebots noch an einer Stelle ein Werbebanner unterbringen, beispielsweise über dem Inhalt. Dann muss also der Frame, in dem anfangs die Datei start.html geladen wird, in zwei Teile geteilt werden: oben die Werbung, unten der Inhalt. Das klingt doch genauso wie zuvor das Beispiel mit den untereinander liegenden Frames. Und genau das ist hier der Clou: Um Frames zu verschachteln, ersetzen Sie an einer beliebigen Stelle das `<frame>`-Element durch ein `<frameset>`-Element. Dann sieht der Code wie folgt aus:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
<!-- frameset-verschachtelt.html -->
<html>
<head>
<title>Frames</title>
```

```
</head>
<frameset cols="200,*" frameborder="0">
<frame src="navigation.html" name="Navi" />
<frameset rows="50,*" frameborder="0">
<frame src="werbung.html" scrolling="no" />
<frame src="start.html" name="Inhalt" />
</frameset>
</frameset>
</html>
```

Fehlt abschließend nur noch das Dokument werbung.html, das Sie im Folgenden abgedruckt finden:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
        <!-- werbung.html -->
        <html>
        <head>
        <title>Werbung</title>
        </head>
        <body>
        <p style="font-weight: bold;">Kaufen! Kaufen! Kaufen!</p>
        </body>
        </html>
```

Noch einmal zusammengefasst, was in dem Code frameset-verschachtelt.html passiert:

- Durch `<frameset cols="...">` wird der Browserbereich in zwei Bereiche aufgeteilt, einer links, einer rechts.
- Im linken Bereich wird die Datei navigation.html geladen.
- Im rechten Bereich wird mit `<frameset rows="">` ein weiteres Frameset geladen, das genau diesen Bereich in einen oberen und einen unteren Bereich aufteilt.
- Im rechten oberen Bereich wird die Datei werbung.html geladen.
- Im rechten unteren Bereich wird die Datei start.html geladen.

Sie sehen also: Gar nicht so schwer, wenn man einmal einen Frame erstellt hat.

Frames verschönern

Abschließend ist es noch an der Zeit für ein paar kleine Verschönerungsmaßnahmen. Zunächst einmal fällt auf, dass der Balken zwischen den **Frames** mit der Maus verschoben werden kann. Der Benutzer kann also die Größen der Frames verändern und damit möglicherweise das Layout ruinieren.

Zum Glück kann das verhindert werden. Das geht nicht mit CSS, aber mit HTML: Verwenden Sie das

Attribut `noresize`. Allerdings muss noch geklärt werden, wo das hingehört. Die Antwort: in das `<frame>`-Element. Zunächst klingt
das etwas merkwürdig, doch mit etwas Nachdenken wird es klar. Stellen Sie sich vor, Sie haben drei Frames innerhalb eines Framesets. Nun möchten Sie aus irgendwelchen Gründen, dass der Balken zwischen dem ersten und dem zweiten Frame verschoben werden
darf, der zwischen dem zweiten und dem dritten Balken allerdings nicht. Es muss also pro Framebalken festgelegt werden, wie es sein soll. Dafür gibt es die folgenden Regeln:

- Wenn Sie in einem Frame `noresize` setzen, können beide Framegrenzen (links und rechts bzw. oben und unten) nicht mehr per Maus verschoben werden.
- Wenn Sie `noresize` nicht setzen, können beide Framegrenzen (links und rechts bzw. oben und unten) per Maus verschoben werden.
- Wenn in einem der an den Balken grenzenden Frames `noresize` gesetzt wird, im anderen nicht, "gewinnt" `noresize` und der Balken kann nicht per Maus verschoben werden.

Im nachfolgenden Code steht `noresize` bei einem der beiden Frames; der Balken dazwischen kann danach nicht mehr verschoben werden:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
<!-- frameset-noresize.html -->
<html>
<head>
<title>Frames</title>
</head>
<frameset cols="200,*">
<frame src="navigation.html" name="Navi" noresize="noresize" />
<frame src="start.html" name="Inhalt" />
</frameset>
</html>
```

Apropos Verschönerungsmaßnahmen: Die beiden Frames sind ja durch einen etwas unschicken grauen Balken voneinander getrennt. Mit ein klein wenig **HTML** ist das allerdings kein Problem: Das Attribut `frameborder` aktiviert oder
entfernt den Rahmen, je nachdem, welchen Wert Sie angeben:

- Beim Wert "1" etwa erscheint der Balken - das ist der Standard.
- Beim Wert "0" dagegen verschwindet der hässliche Rahmen.

Info Zwar sind mit etwas CSS auch andere Effekte möglich (etwa `border-color` für die Farbe des Rahmens), aber auf modernen Websites mit Frames sind praktisch nirgends mehr Rahmen anzutreffen. Also: Weg damit!

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
<!-- frameset-rahmen.html -->
<html>
<head>
<title>Frames</title>
</head>
<frameset cols="200,*" frameborder="0">
<frame src="navigation.html" name="Navi" noresize="noresize" />
<frame src="start.html" name="Inhalt" />
</frameset>
</html>
```

Jetzt ist der Rahmen endlich verschwunden. Abschließend ist noch zu klären, was zu tun ist, wenn innerhalb eines Frames so viel Inhalt steht, dass Scrolling notwendig ist. Die Inhalte haben nicht mehr genug Platz, deswegen erscheint ein Scrollbalken.
In der Regel ist dieses Verhalten gewünscht. Allerdings nicht immer, beispielsweise bei der Navigation, wo ein Scrollbalken einfach störend wäre. Für diesen Fall können Sie im `scrolling`-Attribut (des `<frame>`-Elements) angeben,
wie so etwas gehandhabt werden soll:

- `"auto"`: Der Browser entscheidet automatisch, ob Scrollbalken notwendig sind oder nicht (das ist das Standardverhalten).
- `"no"`: Keine Scrollbalken.
- `"yes"`: Überall Scrollbalken.

So einfach das klingt, so trickreich ist das in der Umsetzung in die Praxis. `"auto"` funktioniert immer, auch `"no"` wird von den Webbrowsern unterstützt. Nur bei `"yes"` scheiden sich die Geister bzw. Webbrowser. Der
Internet Explorer zeigt nur senkrechte Scrollbalken an (also keine waagerechten), der Firefox ignoriert `"yes"` komplett. Aber in der Praxis ist das auch nicht gerade relevant, denn wie häufig möchten Sie waagerechte und senkrechte Scrollbalken
haben, auch wenn der Inhalt der zugehörigen Webseite breit und hoch genug ist. Im folgenden Code sehen Sie, dass bei beiden Frames das Scrolling deaktiviert worden ist. Beim Verändern der Fenstergröße wird das im rechten Frame spürbar. Wie bereits erwähnt,
macht das hauptsächlich in Frames für Navigation oder Werbung Sinn, aber nicht in Inhaltsframes.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
<!-- frameset-scrolling.html -->
<html>
<head>
<title>Frames</title>
</head>
```

```
<frameset cols="200,*" frameborder="0">
<frame src="navigation.html" name="Navi" scrolling="no" />
<frame src="start.html" name="Inhalt" scrolling="no" />
</frameset>
</html>
```

Mehrfenster-Technik Frames

Als der damals führende Browser-Anbieter Netscape 1995 die Technik der **Frames** implementierte und dazu eine Reihe nicht standardisierter HTML-Elemente einführte, wurde das neue Feature von der Gemeinde der Webdesigner begeistert aufgenommen.
Endlich war es möglich, beispielsweise links eine fixe Navigation zu haben, während sich rechts der scrollbare Inhalt befand. Die Navigation befand sich in einer separaten Datei und musste nicht in jeder Datei neu notiert werden.

So weit ist der Segen der Mehrfenstertechnik gut nachvollziehbar. Dennoch tat sich das W3-Konsortium schwer damit, die Frames in den HTML-Standard mit aufzunehmen. 1998 übernahm man das Konzept zwar in HTML 4.0, führte dazu aber eine eigene HTML-Variante
neben `"strict"` und `"transitional"` ein: die Variante `"Frameset"`.

> Mittlerweile gelten Frames in professionellen Kreisen als verpönt. Dafür gibt es mehrere Gründe:

- Inhaltsseiten haben selbst keine vollständige Struktur und in der Regel keine Navigationslinks.
- Anwender können z.B. keine Bookmarks auf bestimmte Unterseiten eines Angebots setzen, weil in der Adresszeile des Browsers immer nur der URI der Seite angezeigt wird, die das Frameset enthält. Durch etwas Programmieraufwand ist dieses Problem zwar
  lösbar, doch die meisten Anbieter, die Frames einsetzen, kümmern sich nicht darum.
- Frames sind ungünstig für Suchmaschinen. Die meisten Suchmaschinen lösen zwar die Frame-Struktur auf und indexieren die inhaltstragenden Seiten. Doch die Links bei Suchtreffern führen dann direkt zu den Unterseiten. Da diese aber nur einen Teil der
  Seite darstellen und meist keine eigene Navigation enthalten, landen die Seitenbesucher in einer Hypertext-Sackgasse, die oft auch vom Layout her wie eine "halbe Sache" wirkt. Der Besucher erhält keinen besonders guten Eindruck vom Anbieter der
  Seite.
- Sind Frames bei der Bildschirmanzeige noch praktisch, so stellen sie beim Ausdrucken von Webseiten nicht selten ein Problem dar. Hat der Anwender den Fokus im Frame für die Navigation, kann es beispielsweise passieren, dass der Browser nur die Navigation
  ausdruckt und nicht wie gewünscht den Inhalt. Unerfahrene Anwender sind dadurch schnell irritiert.
- Bei drei und mehr Frames soll pro Link nicht selten mehr als ein anderer Frame seinen Inhalt ändern.

Mit HTML allein ist es jedoch nicht möglich, zwei oder mehr URIs pro Link zu adressieren. Deshalb wird zu JavaScript gegriffen, was aber nicht funktioniert,
wenn der Anwender JavaScript ausgeschaltet hat. Die gesamte Navigation funktioniert dann nicht mehr.

- Frame-basierte Seiten sind ein Problem für Browser oder Ausgabegeräte, die nicht frame-fähig sind. Zwar erkennen modernere nicht grafische Browser wie Lynx Frames, doch das Handling für den Anwender ist meist unbefriedigend.
- Frames erfordern mehr Kommunikation zwischen Browser und Webserver, da pro Frame Ressourcen übertragen werden müssen.
- Frames verlocken zu einem unfairen, urheberrechtlich sehr bedenklichen Trick: Nur die Navigation gehört zur eigenen Site, während die Inhalte aus fremden Quellen stammen. In der Adresszeile des Browsers ist das für den Anwender jedoch nicht erkennbar.

Im Grunde gibt es zwei Hauptgründe, weswegen Frames immer noch zum Einsatz kommen:

- Bestimmte Inhalte scrollen nicht mit, z. B. eine Navigation, ein Logo usw.
- Wiederkehrende Teile wie der Code für eine Navigation muss nur einmal in einer separaten Datei notiert werden, aber nicht mehr in jeder einzelnen Datei.

Für "non-scrolling-regions" steht mit der CSS-Angabe `position:fixed` mittlerweile eine Alternative zur Verfügung. Wir werden in in anderem Artikel auch zeigen, wie Sie diese Alternative richtig einsetzen.

Gegen das Argument der einmaligen Notation von Code

wird von Frames-Gegnern in der Regel angeführt, dass es serverseitige Möglichkeiten gebe, Code nicht immer wieder kopieren zu müssen. Dies läuft jedoch auf dynamisch erzeugte Seiten hinaus. Im einfachsten Fall über Server Side Includes, meist
aber über Scriptsprachen wie PHP. Wenn Sie ohnehin eine Site entwickeln, die nicht mit statischen HTML-Dateien, sondern etwa mit PHP-Scripts arbeitet, dann sind Frames ohnehin nur lästig und problematisch. Wenn Sie ein **HTML**-basiertes
Projekt dagegen unter anderem auch auf Wegen wie CD-ROM verbreiten möchten, müssen Sie statische HTML-Dateien verwenden. In diesem Fall aber zieht das Argument der Frames-Gegner nicht und es ist unter Umständen durchaus vertretbar, sich bewusst für den
Einsatz von Frames zu entscheiden.

Verweise bei Frames

Die Frame-Datei sieht im Browser auch ganz gut aus; der positive Ersteindruck schlägt aber in (berechtigten) Ärger um, wenn Sie versuchen, auf einen der Links im linken **Frame** zu klicken. Die gewünschte Seite wird zwar geladen, aber ...
ebenfalls im linken Frame. Nach einem Klick auf den Link "Produkte" in der Navigation wird die Produktseite angezeigt, aber leider am falschen Ort.

Auch das lässt sich beheben. Dem Webbrowser ist kein Vorwurf zu machen, denn der kann ja nicht ahnen, in welchem Frame der Link angezeigt werden soll. Deswegen geht der Webbrowser nach einem ganz einfachen Prinzip vor: Der Link wird im aktuellen Frame
(oder Fenster) geöffnet.

Beim Artikel Webseiten mit Text und Links konnten Sie mit dem `target`-Attribut festlegen, ob sich das Linkziel im selben oder in einem neuen Fenster öffnet.

Als Attributswert gibt es die folgenden Möglichkeiten:

- `_blank` - in einem neuen Fenster
- `_self` - im aktuellen Fenster
- `_top` - bei Frames im kompletten Anzeigebereich des Webbrowsers (löst also die Frame-Struktur auf)
- `_parent` - bei Frames im übergeordneten Frameset (bei Verschachtelung)

Es gibt noch einige weitere, selten verwendete Möglichkeiten (die auch nicht von allen Browsern unterstützt werden). Die pfiffigste Möglichkeit jedoch ist es, einen Frame-Namen anzugeben. Wenn Sie das tun, öffnet der Browser das Linkziel in dem Frame
mit dem entsprechenden Namen. Jetzt wissen Sie auch, wozu das `name`-Attribut im `<frame>`-Tag gut ist. Also, an die Tat: Passen Sie die Datei navigation.html an und setzen Sie bei allen Links das `target`-Attribut.
Ein kurzer Blick in die Frameset-Datei klärt auf: Der Inhaltsframe hat den Namen (`name`-Attribut) "Inhalt".

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- navigation.html -->
<html>
<head>
<title>Navigation</title>
</head>
<body>
<h1>Navigation</h1>
<p><a href="produkte.html" target="Inhalt">Produkte</a></p>
<p><a href="kontakt.html" target="Inhalt">Kontakt</a></p>
<p><a href="impressum.html" target="Inhalt">Impressum</a></p>
</body>
</html>
```

Die Datei navigation.html ist bereits dementsprechend angepasst. Und wie Sie im Browser sehen, funktioniert das Ganze jetzt auch wunderbar: Das Linkziel öffnet sich im rechten Frame.
Info Wenn Sie das **Linkziel** für eine komplette HTML-Seite festlegen möchten, können Sie folgendes Element im `<head>`-Abschnitt des Dokuments verwenden:

```
<base target="Inhalt" />
```

Jetzt öffnen sich automatisch alle Links im Frame namens "Inhalt". Soll sich einer der Links in dem Dokument doch auf der aktuellen Seite öffnen, müssen Sie den zuvor erwähnten Wert `"_self"` verwenden.

Links zu anderen Frame-Fenstern

Wie bereits erwähnt, müssen Links innerhalb eines Framesets, deren Ziel in einem anderen als dem aktuellen Frame-Fenster angezeigt werden soll, das gewünschte Frame-Fenster benennen. Anzugeben ist der Name, der bei `<frame ... name=>`vergeben wurde.

```
<a href="index.htm" target="inhalt">Startseite</a>
```

Info Dieser Link öffnet sein Ziel in einem Fenster, das mit `<frame ... name="inhalt">` definiert wurde. Das `target`-Attribut gibt den gewünschten Fensternamen an (**target** = Ziel). Bei Navigationslinks, die alle auf ein anderes Frame-Fenster verweisen, gibt es jedoch noch eine andere Möglichkeit, nämlich im Kopfbereich des HTML-Dokuments ein Zielfenster vorzugeben.

```
<head>
<!-- andere Kopfdatendefinitionen -->
<base target="inhalt">
</head>
```

Info Das `base`-Element kann ebenfalls ein `target`-Attribut enthalten. So notiert wie im Beispiel hat es die Wirkung, dass alle Linkziele des aktuellen Dokuments in einem Frame-Fenster namens `inhalt` geöffnet werden. Groß- und Kleinschreibung werden übrigens unterschieden. Wenn beispielsweise ein Fenster mit `<frame ... name="Inhalt">` definiert wurde, kann es nicht mit `<a ... target= "inhalt">` angesprochen werden! `target`-Angaben
zu nicht existierenden Frame-Fenstern bewirken in den meisten Browsern übrigens, dass der Link stattdessen in einem neuen Fenster oder in einem neuen Tab geöffnet wird. Wichtig zu wissen ist ferner, dass HTML-Dokumente, die in Hyperlinks oder im `base`-Element
das `target`-Attribut benutzen, in der Dokumenttyp-Deklaration nicht mit der HTML-Variante `"strict"` ausgezeichnet werden dürfen. Das `target`-Attribut gehört zu den HTML-Bestandteilen, die im "eigentlich gewünschten"
HTML keinen Platz mehr haben, ebenso wenig wie Frames allgemein. Verwenden Sie für Dokumente, in denen `target`-Attribute vorkommen, stattdessen die HTML-Variante `"transitional"`.

Was sind Frames

Frames haben einige Vorteile. Beispielsweise ist es möglich, die Navigation der Website in einem eigenen **Frame** unterzubringen. Wenn der Benutzer nun auf eine andere Seite navigiert, bleibt der

Navigationsframe unverändert. Das ist natürlich
weniger aufwändig, als auf jeder Unterseite erneut die Navigation unterbringen zu müssen. Außerdem sind Änderungen an der Navigation dann nur in einer Datei erforderlich, nicht mehr in allen.

Allerdings sind Frames auch umstritten. Anfänger im Web haben unter Umständen Schwierigkeiten, einzelne Frames auszudrucken oder ein Bookmark (Lesezeichen) abzulegen. Außerdem kann es Probleme geben, wenn ein Dokument, das in einem Frame abgelegt worden
ist (etwa eine Inhaltsseite), direkt im Webbrowser aufgerufen wird. Dann fehlen nämlich alle anderen, unter Umständen wichtigen Frames, wie etwa ein *Navigationsframe*. Außerdem haben manche Suchmaschinen Frames nicht gerade gerne. Einige (wenige)
Browser, beispielsweise solche für mobile Endgeräte, haben Schwierigkeiten, Frames darzustellen.

> Sie müssen es sich also genau überlegen, ob Frames für Ihre Website geeignet sind oder nicht.

Im Web scheint es zurzeit so zu sein, dass eine spezielle Form von Frames immer größere Verbreitung findet, so genannte Iframes.

Vorbereitungen

Um Frames verwenden zu können, brauchen wir zunächst einige Testdateien. Diese werden wir im Verlauf dieses Kapitels immer weiter anpassen. Zunächst einmal benötigen wir ein Dokument, das drei Navigationslinks enthält:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- navigation.html -->
<html>
<head>
<title>Navigation</title>
</head>
<body>
<h1>Navigation</h1>
<p><a href="produkte.html">Produkte</a></p>
<p><a href="kontakt.html">Kontakt</a></p>
<p><a href="impressum.html">Impressum</a></p>
</body>
</html>
```

Außerdem benötigen wir mehrere Inhaltsseiten: die Start-, Produkt-, Kontakt- und Impressumsseite. Diese sind natürlich hier nur äußerst spärlich mit Inhalt gefüllt, denn es soll primär das Grundprinzip vermittelt werden. Wie Sie diese ansprechender gestalten

können, wurde bereits in den vorherigen Artikeln weitgehend geklärt. Die erste Inhaltsseite ist die Startseite. Diese enthält - weil es gute Sitte ist - noch einmal die wichtigsten Links zu den entsprechenden Unterseiten.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- start.html -->
<html>
<head>
<title>Startseite</title>
</head>
<body>
<h1>Willkommen!</h1>
<p>Herzlich willkommen auf unserer Website! Hier finden Sie Informationen über
uns und unsere Dienstleistungen.
<br>
Nachfolgend einige interessante Links:</p>
<p><a href="produkte.html">Produkte</a></p>
<p><a href="kontakt.html">Kontakt</a></p>
<p><a href="impressum.html">Impressum</a></p>
</body>
</html>
```

Die nächste Inhaltsseite ist die Produktseite. Sie wird später einmal eine Liste aller Produkte enthalten, möglicherweise sogar auf einzelnen Unterseiten. Doch zunächst müssen ein paar magere Informationen ausreichen:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- produkte.html -->
<html>
<head>
<title>Produkte</title>
</head>
<body>
<h1>Produkte</h1>
<p>Wir bieten Ihnen:
<ul>
<li>Ergonomische Schlafkissen aus Silikon</li>
<li>Kuschelige Bettdecken aus Rauhfaser</li>
<li>Bequeme Matratzen aus Styropor</li>
</ul>
</p>
</body>
```

```
</html>
```

Die Kontaktseite enthält zunächst nur die E-Mail-Adresse.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- kontakt.html -->
<html>
<head>
<title>Kontakt</title>
</head>
<body>
<h1>Kontakt</h1>
<p>Sie erreichen uns per E-Mail unter
<a href="mailto:info@schlafshop.xy">info@schlafshop.xy</a></p>
</body>
</html>
```

Abschließend der Pflichteintrag für jede Website: das Impressum. Dort stehen unter anderem der Name des Betreibers und seine Mailadresse.