CSS:
Stylesheet für ein Druck-Layout

Roland Skop

E-Mail: selfhtml@skop.net

Homepage-URL: http://skop.net/

Bei Fragen zu diesem Beitrag bitte den Autor des Beitrags kontaktieren!

Hinweise zum Thema

Um eine optimal formatierte Druckausgabe zu erreichen, kann man serverseitige Programmierung einsetzen, die nicht benötigten Bereiche aus dem HTML-Quelltext entfernt, Zusatzinformationen einfügt usw. Dies ist jedoch nur in Ausnahmefällen für weitreichende Umgestaltungen nötig, da CSS die Möglichkeit bietet, für verschiedene Medientypen unterschiedliche Layouts festzulegen – es ist somit nicht nötig, eine extra HTML-Seite zu erstellen. Mit den im Folgenden vorgestellten Methoden lässt sich das Layout für die Druckausgabe (auch Print-Stylesheet genannt) optimieren.

Dieser Artikel handelt von der Erstellung eines Layouts für die Druckausgabe, dennoch müssen Sie die Beispiele nicht unbedingt ausdrucken, um das Ergebnis zu sehen. Durch Auswahl folgender Menüpunkte und Tastaturkürzel gelangen Sie zur Druckansicht bzw. Druckvorschau Ihres Browsers, die der tatsächlichen Ausgabe entspricht:

- Camino: File / Print (Befehlstaste-P), dort der Button „Preview"
- Firefox 1.x: Datei / Druckvorschau
- Internet Explorer 5.0 Mac: Datei / Seitenansicht
- Internet Explorer 5.x Win: Datei / Seitenansicht
- Internet Explorer 6.0 Win: Datei / Druckvorschau

- Konqueror: Dokument / Drucken ... / [x] Vorschau (aktivieren)
- Mozilla (Netscape 6 und 7): Datei / Druckvorschau bzw. `[Alt][d], [v]`
- Omniweb: File / Print (Befehlstaste-P), dort der Button „Preview"
- Opera: Datei / Druckvorschau bzw. `[p]`, ab Version 8 `[Shift]-[p]`
- Safari: Ablage/Drucken (Befehlstaste-P), dort der Button „Vorschau"

Für den Fall, dass Ihr Browser CSS nicht ausreichend beherrscht (was vor allem für Netscape 4 und den Internet Explorer in allen Versionen gilt), enthält jedes Beispiel einen Screenshot.

Einbindung von Medientypen

Es existieren mehrere Methoden, Stylesheets für verschiedene Medientypen einzubinden. Entweder, Sie binden mehrere externe Stylesheets ein, oder Sie fassen alle Medientypen innerhalb eines Stylesheets zusammen. Die zweite Methode ist zwar übersichtlicher, doch leider nicht sehr zuverlässig. Bei der Einbindung mittels CSS-Syntax (`import url()` bzw. `@media ... {}`) versagt der Internet Explorer 5 jeweils in der Windows- oder der Mac-Version. Netscape 4 streicht an dieser Stelle gänzlich die Segel.

Die HTML-Syntax, pro Medientyp ein Stylesheet per `<link rel= ... >` einzubinden, funktioniert dagegen in den meisten Browsern. Sie wurde daher zur Erstellung der Beispiele herangezogen. Die Einbindung sieht folgendermaßen aus:

Quelltext	Erläuterung
`<link rel="stylesheet" type="text/css" href="selfhtml.css" media="all" />`	Dieser HTML-Tag im `<head>` der Seite bindet ein Stylesheet namens selfhtml.css für alle Medientypen ein. Die darin enthaltenen Definitionen gelten somit immer.
`<link rel="stylesheet" type="text/css" href="screen.css" media="screen" />`	Dieser HTML-Tag im `<head>` der Seite bindet ein Stylesheet namens screen.css für den Medientyp screen ein. Die darin enthaltenen Definitionen gelten somit für die Anzeige auf dem Monitor.
`<link rel="stylesheet" type="text/css" href="print.css" media="print" />`	Dieser HTML-Tag im `<head>` der Seite bindet ein Stylesheet namens print.css für den Medientyp print ein. Die darin enthaltenen Definitionen gelten somit für die Ausgabe auf dem Drucker.

Beachten Sie:

Enthält das allgemeingültige Stylesheet „selfhtml.css" Definitionen, die auf einem Ausdruck störend wären, müssen Sie diese im Druck-Stylesheet „print.css" explizit überschreiben. Es ist daher vorteilhaft, ausschließlich für die Monitor-Ansicht notwendige Definitionen in „screen.css" zu schreiben.

Hinweis:

In den folgenden Quelltext-Beispielen ist der Code für die Einbindung der Stylesheets nicht immer extra angeführt, eine Kommentarzeile weist jedoch darauf hin, zu welchem Stylesheet die Definitionen gehören. Das allgemeingültige Stylesheet von SELFHTML aktuell heißt selfhtml.css.

Beispiele

Abweichende Formatierung

Es ist sinnvoll, für den Druck andere Formatierungen einzusetzen, als für die Anzeige auf dem Bildschirm. Dies gilt vor allem für positionierte Bereiche, ebenso für Farbgestaltung und Schriftarten.

Positionierung und Farbgestaltung

Anzeigebeispiel: So sieht's aus

Auszug aus dem Quelltext:

Quelltext	Erläuterung
selfhtml.css	
`h1{ font-size:26px; margin-bottom:18px; }`	Zunächst werden allgemeingültige Eigenschaften definiert, im Beispiel wird die Überschrift ersten Grades formatiert. Dies gilt für alle Medientypen.
screen.css	
`h1 { text-align:right; border-bottom:3px dashed #00f; color:#008; background-color:inherit; }`	Die Gültigkeit aller in dieser Datei notierter Zeilen ist auf den Medientyp `screen` beschränkt, welcher bei der Anzeige auf dem Bildschirm Anwendung findet.
print.css	
`h1, ... { color:#000; background-color:#fff; }`	Im letzen Stylesheet schließlich, print.css, werden alle für den Druck relevanten Definitionen zusammengefasst. Im Beispiel wird den meisten Elementen schwarze Schrift auf weißem Hintergrund zugewiesen.

Beachten Sie:

Die meisten Druckertreiber ermöglichen es, einen Ausdruck in Graustufen zu erstellen, um teure Farbtinte zu sparen. Umgekehrt ist dies nicht möglich – formatieren Sie im Druck-Layout alles in Graustufen, wird der Ausdruck, selbst wenn Ihre Besucher dies wünschen, keine Farben enthalten. Es gilt hier abzuwägen, wie weit Sie Ihren Besuchern entgegekommen wollen, ohne sie zu bevormunden. Weiters ignorieren die meisten Browser beim Druck die von Ihnen definierten Hintergrundfarben und -bilder, um auch an dieser Stelle Farbe bzw. Toner zu sparen. Dies ist sinnvoll und gewollt, lässt sich daher auf normalem Weg nicht umgehen.

Schriftarten mit und ohne Serifen

Serifen sind die abschließenden Striche an Buchstaben. Während auf dem Monitor serifenlose Schriftarten besser zu lesen sind, sollten Sie für die Erstellung des Drucklayouts auf Schriften mit Serifen zurückgreifen.

Anzeigebeispiel: So sieht's aus

Quelltext	Erläuterung
screen.css	
`* {`	Die Ausgabe aller Elemente erfolgt auf dem Monitor mit einer

`font-family:verdana,arial,sans-serif;` `}`	serifenlosen Schrift, ...
print.css	

`* {`
`font-family:"times new roman",times,serif;`
`text-align:justify;`
`}`
	...auf dem Drucker jedoch mit Serifen und im Blocksatz.

Die linken zwei Buchstaben stellen die Schriftart „Verdana" (ohne Serifen), die rechten beiden „Times New Roman" (mit Serifen, rot markiert) dar.

Ausblendung von Elementen

Mittels `display:none` lassen sich Elemente ausblenden, die auf einem Ausdruck nicht benötigt werden, beispielsweise Formulare, aber auch ganze Bereiche, wie etwa die Navigation. Um Seitenbestandteile auszublenden, bieten sich mehrere Möglichkeiten an:

Anzeigebeispiel: So sieht's aus

Quelltext	Erläuterung
print.css	

`#id {`
`display:none;`
`}`
	Der erste Selektor `#id` eignet sich, wenn Bereiche, die nicht gedruckt werden sollen, klar mit einer `ID` abgegrenzt wurden.

`element {`
`display:none;`
`}`
	Der zweite Selektor betrifft Tags, die auf einem Ausdruck nicht benötigt werden, wie etwa Formulare.

`.klasse {`
`display:none;`
`}`
	Der dritte Selektor gilt für alle Elemente, denen eine entsprechende Klasse zugewiesen wurde.

`#id, element, .klasse {`
`display:none;`
`}`
	Diese Methoden lassen sich problemlos kombinieren.

Ebenso ist es möglich, Elemente ausschließlich beim Ausdruck zu berücksichtigen. Weisen Sie dem gewünschten Element im allgemeingültigen Stylesheet `display:none` zu, im Drucklayout „print.css" überschreiben Sie dies mit `display:block|inline|...`.

Beachten Sie:

Textbrowser und Suchmaschinen ignorieren CSS und somit die von Ihnen definierte Ausblendung. Das Dokument sollte daher auch bei gänzlichem Verzicht auf CSS logisch aufgebaut sein. `visibility:hidden` ist an dieser Stelle übrigens nicht geeignet, es bewirkt zwar ebenfalls, dass Elemente nicht angezeigt werden, doch gibt es den für das Element reservierten Raum im Dokument nicht frei, es entstünden daher freie Flächen. `display:none` hingegen entfernt Bereiche vollständig aus dem Elementenfluss – so, als wären sie im Quelltext

nicht vorhanden.

Manueller Seitenumbruch

Wollen Sie einen Seitenumbruch erzwingen oder verhindern, können Sie dies mittels page-break-before sowie page-break-after erreichen.

Hochformat / Querformat

Muss ein relativ breites Layout auf ein Blatt Papier passen, weil eventuell eine Tabelle mit vielen Spalten auf der Seite vorhanden ist, kann man die Ausgabe auf dem Drucker mittels `@page { size:landscape }` im Querformat definieren. SELFHTML enthält eine Beschreibung und ein Beispiel.

Beispiele mit generierten Inhalten

Beachten Sie:

Generierte Inhalte werden generell nur von modernen CSS-fähigen Browsern unterstützt. Die in diesem Artikel angeführten Beispiele funktionieren in Mozilla und Opera Konqueror/Safari?. Der Internet Explorer beherrscht weder generierte Inhalte, noch Attribut-Selektoren, welche ebenfalls verwendet wurden, Netscape 4 scheitert bereits bei der Einbindung. Allerdings kommt es beim Einsatz dieser Techniken in den genannten Browsern zu keinen Fehlern, die Bereiche im Stylesheet werden einfach ignoriert. Dem uneingeschränkten Einsatz in der Praxis steht also nichts im Wege.

Visualisierung des Verweisziels

Beim Ausdruck von Webseiten geht eines der wichtigsten Merkmale von Hypertext verloren – die Verlinkung. Mit generiertem Inhalt ist es jedoch möglich, das Attribut href, welches das Linkziel beinhaltet, auf Papier sichtbar darzustellen:

Anzeigebeispiel: So sieht's aus

Quelltext	Erläuterung
print.css	
`a[href]:after { content:" <"attr(href)">"; color:#888; background-color:inherit; font-style:italic; size:80%; }`	Allen Elementen `a`, welche ein Attribut `href` besitzen, wird der Inhalt dieses Attributs nach einem Leerzeichen und einer spitzen öffnenden Klammer angefügt. Hinter dem Attributwert wird eine schließende spitze Klammer eingefügt. Die restlichen Definitionen entsprechen denen dieses und der folgenden Beispiele und sind frei variierbar.

Visualisierung von Ankernamen

Nicht nur die Verweisziele, sondern auch die im Dokument gesetzten Anker gehen beim Ausdruck normalerweise verloren. Folgende Definition im Stylesheet fügt diese wieder ein:

Anzeigebeispiel: So sieht's aus

Quelltext	Erläuterung
print.css	
`*[name]:after { content:"[#"attr(name)"]"; color:#888; background-color:inherit; font-style:italic; size:80%; }`	Allen Elementen, die ein `name`-Attribut aufweisen uns somit als Anker dienen, wird der Inhalt des Attributs in der Form „[#ankername]“ angefügt.

Hinweis:

Dieses Beispiel eignet sich auch zur Anwendung im Bilschirm-Stylesheet.

Visualisierung von IDs

In XHTML 1.1 ist das `name`-Attribut nicht mehr vorgesehen, es wird durch `IDs` ersetzt. In diesem Fall ersetzt man im Selektor `[name]` durch `[id]` und bei der Eigenschaft `content:` den Wert `attr(name)` durch `attr(id)`. Da der generierte Inhalt damit bei allen IDs im Dokument eingefügt wird, ist es zweckmäßig, den Selektor auf Überschriften zu begrenzen.

Anzeigebeispiel: So sieht's aus

Quelltext	Erläuterung
print.css	
`h1[id]:after, h2[id]:after, ... { content:"[#"attr(id)"]"; color:#888; background-color:inherit; font-style:italic; size:80%; }`	Allen Überschriften, die ein Attribut `id` aufweisen und somit als Anker dienen, wird der Inhalt des Attributs in der Form „[#ankername]“ angefügt.

Visualisierung weiterführender Information des `title`-Attributs

Wenn Sie das `title`-Attribut einsetzen, um Zusatzinformationen per Tooltip zugänglich zu machen, müssen Sie beim Ausdruck nicht darauf verzichten.

Anzeigebeispiel: So sieht's aus

Quelltext	Erläuterung
print.css	

`*[title]:after { content:" ("attr(title)")"; color:#888; background-color:inherit; font-style:italic; size:80%; }`	Allen Elementen, die ein Attribut `title` aufweisen, wird der Inhalt dieses Attributs angefügt. Der generierte Inhalt besteht aus einem Leerzeichen, einer öffnenden runden Klammer, dem Attribut-Inhalt und einer schließenden runden Klammer: „ (title-Attribut)“

Hinweis:

Mit dieser Methode lassen sich auch Abkürzungen wie **abbr** und **acronym**, aber auch Bilder, Verweise usw. mit einem zusätzlichen Erklärungstext versehen.

Visualisierung von Bildbeschreibungen

Das `alt`-Attribut

Innerhalb des bei Bildern zwingend vorgeschriebenen `alt`-Attributs ist ein alternativer Text einzufügen, der den Inhalt des Bildes beschreibt. Wollen Sie diesen im Drucklayout zusätzlich berücksichtigen, können Sie folgende Definitionen verwenden:

Anzeigebeispiel: So sieht's aus

Quelltext	Erläuterung
print.css	
`img[alt]:after { content:" ("attr(alt)")"; }`	Der Inhalt des `alt`-Attributs wird nach dem Bild in runden Klammern angezeigt: „ (alt-Attribut)“

Wollen Sie den generierten Inhalt unter den Bildern anzeigen, können Sie mit „`\A` “ im Wert der Eigenschaft `content:` einen Zeilenumbruch einfügen:

Quelltext	Erläuterung
print.css	
`img[alt]:after { content:"\A ("attr(alt)")"; }`	„`\A` “ erzeugt den gewünschten Absatz.

Das `longdesc`-Attribut

Das Attribut **longdesc** beinhaltet die Adresse einer Quelle, an der sich eine längere Bildbeschreibung abrufen lässt. Textbrowser zeigen an dieser Stelle einen entsprechenden Verweis an. Folgendermaßen wird dieses Attribut sichtbar in das Dokument eingefügt:

Anzeigebeispiel: So sieht's aus

Quelltext	Erläuterung
print.css	

`img[longdesc]:after { content:"\A (Beschreibung: "attr(longdesc)"); }`	Der Inhalt des Attributs `longdesc` wird nach einem Zeilenumbruch und einem Leerzeichen in runden Klammern in das Dokument geschrieben: „ (Beschreibung: longdesc-Attribut)“

Visualisierung von Zitat-Quellen und -Zeichen

Das für kurze Zitate vorgesehene Element `q` und das für längere Zitat-Blöcke bestimmte Element `blockquote` lassen sich mit einem Attribut `cite` versehen, das eine Quelle in Form des URI beinhaltet. Mit generiertem Inhalt lässt sich der Attribut-Inhalt visualisieren:

Anzeigebeispiel: So sieht's aus

Quelltext	Erläuterung
print.css	
`*[cite]:after { content:close-quote" (Quelle: "attr(cite)"); font-size:80%; }`	Hinter jedem Element, das ein Attribut `cite` aufweist, wird die Quelle des Zitats in Form des URI angefügt: „ (Quelle: http://example.com/) “

Beachten Sie:

Die „Gänsefüßchen“, die Sie auf Ihrer Tastatur finden, entsprechen nicht der deutschen Typographie. Korrekt wäre die Verwendung von „ “ als doppelte und , ‘ als einfache Anführungszeichen. Auch dies lässt sich mit generiertem Inhalt erreichen:

Quelltext	Erläuterung
print.css	
`* { quotes: "\201E" "\201C" "\201A" "\2018"; }`	Zunächst werden allen Elementen die korrekten Anführungszeichen zugewiesen. Diese wurden in Unicode definiert.
`q:before, blockquote:before { content:open-quote; }`	Anschließend wird allen Tags `q` und `blockquote` die öffnende Variante vorangestellt...
`q:after, blockquote:after { content:close-quote; }`	...und die schließende angefügt.

Der Artikel Typographie für Webautoren beschäftigt sich ausführlich mit diesem Thema.

Manipulation von Kopf- und Fußzeilen

Die Einstellungen bezüglich des Andrucks von Kopf- und Fußzeilen obliegen dem Benutzer des Browsers. Von Seiten des Erstellers lassen sich diese mit CSS2 nicht beeinflussen. Dies wird vermutlich mit CSS3 möglich sein.

Unterstützung der in diesem Artikel vorgestellten Methoden durch folgende Browser:

Syntax:	Netscape 4	MSIE 5.0, 5.5, 6.0 Windows	MSIE 5.0 Mac	Konqueror	Safari	Firefox, Mozilla, Netscape 6, 7	Opera 6 - 9
`<link rel="" media="">`	fehlerhaft	ja	ja	ja	ja	ja	ja
`@media ...`	nein	ja	nein	ja	ja	ja	ja
`@import url("bla.css")`	nein	nein	ja	ja	ja	ja	ja
`*[attribut]`	nein	nein	nein	ja	ja	ja	ja
`:before`	nein	nein	nein	nein	nein	ja	ja
`:after`	nein	nein	nein	nein	nein	ja	ja

Weiterführende Links

Die folgenden Stellen werden empfohlen, um das obige Beispiel besser zu verstehen, oder um weitere Möglichkeiten und Details zu erfahren.

- Tipps und Tricks: Automatische Nummerierung
- CSS2 Media Tests
- W3C: Beschreibung der Pseudoelemente `:before` und `:after`
- Typographie für Webautoren
- Typographie - Beschreibung von Serifen

SELFHTML aktuell Artikel CSS