

Korrekturen, Erläuterungen und Ergänzungen zu „**Cascading Stylesheets**“ von *Dan Shafer* und *Kevin Yank*

erschienen im dpunkt.verlag
1. Auflage 2004
ISBN 3-89864-248-8

Achtung: Bei diesem Dokument handelt es sich **nicht** um eine Publikation des dpunkt.verlages!

Vorwort

Auf dem deutschen Büchermarkt gibt es mittlerweile zahlreiche Bücher zur CSS-Technologie. Ein Buch das u. a. in gewisser Weise heraussticht ist „Cascading Stylesheets“ von Dan Shafer und Koauthor Kevin Yank. Es ist der interessante und sinnvolle Ansatz, die Sprache anhand eines ausgereiften Projektes zu vermitteln. Es hat etwas Spannendes, damit zu arbeiten und zu beobachten, wie mit CSS eine tabellenlose Web-Site entsteht.

Leider musste ich beim Lesen feststellen, dass die erste Hälfte des Buches an sehr vielen Stellen fehlerbehaftet ist. Es fängt an bei häufig falsch und widersprüchlich eingesetzter Terminologie, über nicht nachvollziehbaren Auslassungen bis hin zu stellenweise gravierend falschen Aussagen. Vielleicht liegt es an der deutschen Übersetzung des Buches aus dem Englischen, ich kenne das Original nicht und kann das daher nicht beurteilen. Wie dem auch sei, der zweite Teil des Buches ist auf jeden Fall besser geschrieben.

Für mich war das Buch auf jeden Fall stellenweise sehr verwirrend und unnötig kompliziert geschrieben. Ganz zu schweigen von den zahlreichen falschen Aussagen. Irgendwann habe ich dann angefangen, die Stellen im Buch, die verwirrend oder falsch geschrieben sind, zu korrigieren und zusätzliche Erläuterungen zu schreiben. So ist diese 18 Seiten umfassende Arbeit entstanden.

Der Verlag selbst bietet zwar auf seiner Web-Site ein Dokument namens „Korrekturen und Ergänzungen“ an, doch das ist inhaltlich ziemlich dünn besetzt und beschränkt sich letztlich auf ein paar Quelltextfehler. Runterladen sollte man es sich trotzdem, da es natürlich nicht Bestandteil dieser Arbeit ist. Hier die Web-Adresse:

<http://www.dpunkt.de/leseproben/3-89864-248-8/Korrekturen%20und%20Ergaenzungen.pdf>

Natürlich kann auch ich Fehler machen, obwohl auch ich große Sorgfalt habe walten lassen. Bei fast allen meiner Korrektur- bzw. Verbesserungsvorschläge berufe ich mich auf die CSS2-Spezifikation des W3-Konsortiums, die ja höchsten Verbindlichkeitscharakter für die Terminologie und Anwendung hat:

<http://www.w3.org/TR/1998/REC-CSS2-19980512>

Ich habe sehr viele Stunden damit verbracht, diese Arbeit zu verfassen. Ich hoffe, dass die von mir zusammengetragenen Korrekturen, Vorschläge und verfassten Erläuterungen hilfreich sind, Ihnen Zeit sparen und Ihr Verständnis vertiefen.

Für Anregung und Kritik bin ich dankbar.

Augsburg, im Juni 2004

Ali Reza Razavi

E-Mail: razavi@comvidya.de

Hinweis: Alle Angaben in dieser Arbeit wurden von mir mit größter Sorgfalt erarbeitet. Die Arbeit erhebt aber keinen Anspruch auf Richtigkeit oder Vollständigkeit. Für eventuelle Schäden, die im Zusammenhang mit der Verwendung dieser Arbeit stehen, übernehme ich keine Haftung.

Copyright © Ali Reza Razavi – Alle Rechte vorbehalten.

Eine Verwertung dieser Arbeit, die über übliche Zitate hinausgeht, bedarf meiner ausdrücklichen Genehmigung.

Hauptteil

Grundsätzlich ist die Korrektur folgenderweise aufgebaut:

Seite, Position und Inhalt, wobei Zitate aus dem Buch *kursiv formatiert* sind.

Die jeweilige Position gebe ich mit „**Oben**“, „**Mitte**“ oder „**Unten**“ an. Das heißt, dass sich die Aussage, auf die ich mich beziehe, **im oberen, mittleren oder unteren Drittel der Seite** befindet. Ich verzichte bewusst auf Zeilenangaben, da es sowohl für mich, als auch für den Leser einfacher sein dürfte, sich grob auf der Seite zu orientieren, anstatt Zeilen zu zählen.

S. 8 Mitte

Es ist keine besonders gute Idee, hier den Begriff „Stile“ als „*Anweisungen für die Darstellung von HTML-Elementen*“ zu verwenden. Wenn Stile Anweisungen wären, was wäre dann eine Stilanweisung von der ansonsten die Rede ist? Ein Stil hat meiner Ansicht nach etwas mit der Formatierung bzw. Darstellung zu tun.

Anweisungen zur Formatierung von Elementen, Stilanweisungen also, werden in CSS Regeln genannt. Somit ist ein Stil eine Sache, eine Stilanweisung eine andere, wie man später im Buch dann auch hier und da lesen kann, z.B. auf Seite 14 unten oder 47 unten. Ein Stil ist die Darstellung bzw. das Format eines Elements oder mehrerer Elemente und wird durch Regeln definiert.

S. 10 Oben

„Jeder Stil, egal ob in einem Stylesheet eingebettet oder nicht, besteht aus einer oder mehreren Regeln.“

Falls ein Stil eine Anweisung wäre, wie auf Seite 8 Mitte behauptet, dann bestünde sie also aus einer oder mehreren Regeln. Eine Regel ist jedoch eine CSS-Anweisung. Somit bestünde ein Stil, also eine vermeintliche Anweisung zur Darstellung von HTML-Elementen, aus einer oder mehreren Anweisungen. Das ist einfach höchst verwirrend und falsch. Andererseits wäre hier die Beschreibung von „Stil“, wenn man so will, richtig, falls damit „Stylesheet“ gemeint sein sollte. Doch das stünde eben im Widerspruch zur Beschreibung von S. 8 Mitte. Übrigens werden Stile nicht eingebettet, sondern Stylesheets in einem HTML-Dokument.

Kurze Zusammenfassung:

Ein Stylesheet besteht aus mindestens einer Anweisung zur Darstellung von Elementen bzw. HTML-Dokumenten. In CSS nennt man eine solche Anweisung Regel, die wiederum aus einem Selektor und mindestens einer Deklaration besteht. Regeln kann man auch als Stilanweisungen bezeichnen. Ein Stil ist die Darstellungsform bzw. die Formatierung von HTML-Elementen.

S. 11 Mitte

„In diesem Fall bezieht sich der Selektor auf die Schriftfamilie, in der der Text dargestellt werden soll.“

Wie kann das sein? Ein Selektor bezieht sich nicht auf CSS-Eigenschaften, sondern auf Elemente, für die CSS-Stileigenschaften definiert werden sollen.

S. 16 Mitte

„...der mit den eingebetteten Stylesheets diese Seite erzeugt.“

Diese Aussage ist irreführend. Eigentlich wird hier lediglich ein einziges Stylesheet eingebettet.

S. 33 Oben

„Ich habe aber auch auf die Gefahren der geschilderten Lösungen hingewiesen.“
Leider nicht, denn auf Seite 31 heißt es *„Dieser Weg birgt einige Gefahren, die wir später näher betrachten werden.“*

S. 43 Oben

„Ein Rückblick auf die drei Methoden, HTML-Elemente mit CSS zu definieren.“
Gemeint ist wohl mit CSS Stile für HTML-Elemente zu definieren.

S. 47 Oben

„Erstens hängt der Einsatz eines CSS-Selektors davon ab, welche Stellung er in der Hierarchie eines Dokuments einnimmt.“

Schwer verdaulicher Satz, denn wie kann der Einsatz eines Selektors von seiner Stellung in der Hierarchie eines Dokuments abhängen? Ein Selektor kann ja keine Stellung in der Dokumenthierarchie einnehmen, geschweige denn sein Einsatz davon abhängig sein. Besser ist wohl *„Die Anwendung einer Stilregel ist abhängig von der Übereinstimmung seiner Selektorstruktur mit den Elementen des Dokumentbaums.“*

S. 47 Mitte

„Zweitens nehmen Elemente, für die nicht extra alle Eigenschaften neu definiert werden, die Eigenschaftsdefinitionen ihres direkt übergeordneten Elements an.“
Besser wäre, *„die Eigenschaftsdefinitionen ihrer Vorfahren an“.*

S. 49 Tabelle

Bei Universeller Selektor: Laut Spezifikation muss der universelle Selektor angegeben werden, es sei denn er ist nicht die einzige Komponente des Selektors.
Bei Elementtyp: Es sollte heißen: *„Stilregel ist gültig für alle im Selektor genannten HTML-Elementtypen“.*

Bei Pseudo-Klassen-Selektor: Pseudo-Klassen verändern nicht ihren Zustand durch Interaktion mit dem Benutzer, wie auch später auf Seite 53 richtig geschrieben wird. Sie können aber „dynamisch“ sein (so bezeichnet das W3C das leider irreführenderweise auch), d.h. ein Element kann eine Pseudo-Klasse erhalten oder auch nicht, je nach seinem Zustand durch die Interaktion mit dem Benutzer. Das ist das, was sich verändert, die zugewiesene Klasse eines Elements je nach seinem Zustand, nicht die Klasse selbst. Mit ihnen definiert man also lediglich (Pseudo-)Klassen für bestimmte Zustände, in die ein Element eintreten kann.

S. 50 Oben

„Um einen Stil für mehrere HTML-Elemente auf einmal zu definieren benutzt man einen Klassenselektor.“

Besser wäre: *„Um einen Stil für bestimmte HTML-Elemente zu definieren...“*

S. 51 Oben

„Eine Stilregel mit ID-Selektor überschreibt sämtliche andere CSS-Eigenschaftsdefinitionen, die für ein HTML-Element gelten.“

Diese Aussage ist äußerst missverständlich, denn ein ID-Selektor hat lediglich Priorität vor Attribut- und Klassen-Selektoren. Wird in zwei Stilregeln, von denen eine einen ID-Selektor hat, jeweils dieselbe CSS-Eigenschaft mit unterschiedlichen

Werten definiert, so hat eben die Stilregel mit dem ID-Selektor Priorität vor der Stilregel mit beispielsweise einem Attribut-Selektor.

„Folgende Stilregel definiert ein Element mit der ID einzigartig:“

Es müsste wohl heißen: „Folgende Stilregel definiert einen Stil für ein Element mit dem ID-Attribut einzigartig.“

„Auch ein ID-Selektor ist nur für Elemente gültig, die mit dem Namen des Selektors in einem HTML-Attribut gekennzeichnet sind.“

Besser wäre: „Auch ein ID-Selektor gilt nur für ein Element, dessen ID-Attribut-Wert dem des Selektors entspricht.“

S. 52 Mitte

„Programmieraufwand“

Nur nebenbei bemerkt: hier entsteht der Eindruck, dass CSS programmiert werden würden, dabei werden sie einfach nur geschrieben.

S. 52 Unten

first-child ist kein Pseudo-Element, sondern **eine Pseudo-Klasse**.

S. 53 Oben

„Das Pseudo-Element first child...bezieht sich immer auf die direkten Nachfahren des HTML-Elements.“

Das ist nicht richtig. first-child bezieht sich nur auf **den ersten direkten Nachfahren** eines HTML-Elements, also auf das erste Kind-Element (= first-child).

S. 54 Oben

Die Pseudo-Klasse :lang() bestimmt nicht die Einstellung des Sprachattributes lang für ein HTML-Element (wie denn auch?). Mit ihr wird eine (Pseudo-)Klasse für Elemente mit einer bestimmten Wertzuweisung an das HTML-Universal-Attribut lang (eventuell auch noch abhängig vom Meta-Element) definiert.

S. 55 Mitte

„Selektor für direkt über-und untergeordnete Elemente“ ist doppeldeutig (sowohl direkt übergeordnete Elemente als auch untergeordnete?) und irreführend. In der CSS-Spezifikation heißt es einfach Kind-Selektor, was auch näher liegender ist. Außerdem heißt es hier: *„...die einem anderen Dokument in der Vererbungskette direkt(!) nachfolgen.“* Natürlich ist anstatt Dokument Element gemeint.

Weiter unten auf der Seite heißt es: *„Den Selektor sollte man von daher nur dann verwenden, wenn eine Seite absichtlich nicht mit dem IE für Windows dargestellt werden soll.“* Diese Aussage ist nicht ganz richtig. Denn durch den Selektor allein ergibt sich nicht, ob eine Seite dargestellt wird oder nicht. Es müsste z.B. heißen: *„...wenn ein Stil absichtlich nicht mit dem IE dargestellt werden soll.“*

S. 56 Mitte

„Selektoren für benachbarte Elemente“: was hier nicht deutlich beschrieben wird ist, dass der **zweite Teil des Selektors**, also h2, der Elementtyp ist, auf den sich die Deklaration bezieht.

Außerdem ist weiter unten auf der Seite und auf der nachfolgenden Seite von „Stilregeln, die gültig sind“ die Rede. „gilt“ wäre wohl ein besserer Ausdruck, da ja der Syntax einer Stilregel etwas über ihre Gültigkeit aussagt. Nicht aber die verschiedenen (erlaubten) Selektoren.

S. 56 Unten

„Jeder Attribut-Selektor bestimmt auf seine spezifische Art, dass eine Stilregel nur auf Elemente angewandt wird...“

Eine Stilregel besteht ja selbst aus einem Selektor und einer Deklaration mit Eigenschaften. Der Selektor ist also Teil der Stilregel selbst und bestimmt somit, ob die in der Deklaration angegebene **Stil-Eigenschaft** angewendet wird. Genau genommen müsste es so heißen.

S. 57 Mitte

„Mit der dritten der oben aufgeführten Varianten...werden Attribute angesprochen, denen ein bestimmter Wert zugewiesen wird.“

Es werden mit Attribut-Selektoren keine Attribute angesprochen, sondern, wenn überhaupt, Elemente, deren Attribute einen bestimmten Wert haben. Eigentlich werden Elemente bestimmt, deren Attribute einen bestimmten Wert haben.

S. 58 Fußnote (und S. 59 Oben)

„In unserem Kontext bezeichnen wir mit relativ ein Verhältnis zu einem Wert in einer Stilregel oder einem HTML-Attribut.“

Diese Beschreibung von „relativ“ ist ziemlich abstrakt und allgemein. Auch auf Seite 59 oben heißt es: *„Mit einer relativen Maßangabe weist eine Stilregel den Browser an, die Größe der Schrift oder eines anderen HTML-Elements relativ zur Größe eines Ausgangswertes zu bestimmen.“* Denn es ist stets **ein bestimmter Wert** auf den sich ein relativer Wert bezieht, nämlich **die Schriftgröße** (bzw. die x-Höhe der Schrift) des Elements selbst. Eine Ausnahme stellt eine relative Angabe bei font-size dar, denn da bezieht sich ein relativer Wert auf die Schriftgröße des Elternelements. Außerdem heißt es *„...die Größe der Schrift oder eines anderen HTML-Elements..“*, gemeint ist aber sicher *„...oder einer anderen CSS-Stil-Eigenschaft.“*

Hinzu kommt, dass das vom Autor beschriebene Beispiel etwas unglücklich ausgewählt ist, zumal es sich bei einem Prozentwert laut Spezifikation eigentlich nicht um einen relativen Wert handelt (obwohl er dennoch einen relativen Bezug hat):

„Das folgende Beispiel zeigt eine Stilregel, durch die die Schriftgröße für alle in Absätzen ausgezeichneten Textstellen 150% des Standardwertes betragen sollen.“

Es wird dabei auch nicht klar, dass es sich bei dem „Standardwert“ um die Schriftgröße des Elternelements von p handelt. Darauf bezieht sich nämlich **in diesem Fall** der Prozentwert.

Bitte auch hier weiter lesen:

S. 60 Oben

„Pixelangaben verhalten sich dagegen komplett anders. Verwendet man Pixel als Maßeinheit für die Schriftgröße, wird ein stabiles Größenverhältnis zwischen Text und Bildern erzeugt, das auch bei verschiedenen Grafikauflösungen oder Monitorgrößen erhalten bleibt.“

Absolute Maßangaben machen nur bei Ausgabegeräten Sinn, deren physische Eigenschaften bekannt sind. Das heißt, dass sie für die Gestaltung von Web-Seiten

eher nicht in Frage kommen, da eine Web-Site ja in vielen verschiedenen Umgebungen angezeigt wird. Der Browser muss ja eine absolute Angabe in Pixel umrechnen. Zumindest unter Windows ist dieser Umrechnungsfaktor eine Konstante, die offenbar nicht von der Bildschirmauflösung abhängt. Folglich bleibt zumindest unter Windows auch bei absoluten Maßangaben das Verhältnis zwischen Text und Bildern bei unterschiedlichen Auflösungen bestehen.

„Erstens sind sie nach wie vor die meistangewandte absolute Maßeinheit in der Entwicklung von Webseiten.“

Selbst wenn Pixel die am häufigsten verwendete Maßeinheit sein sollten, ist es doch noch kein Grund, sie deswegen häufig einzusetzen, oder?

Auch das nächste Argument hinkt etwas, da auch Schriftgrößen mit relativen Maßangaben nicht „intelligent“ reagieren und bei unterschiedlichen Auflösungen unterschiedlich groß dargestellt werden.

S. 61 Unten

„% steht für einen Prozentwert von der Ausgangshöhe der Schrift.“

Die Überschrift des Kapitels ist: *„Maßangaben / Relative Werte“*. Von daher ist es irreführend allgemein zu behaupten, dass sich eine Prozentangabe auf die Ausgangshöhe einer Schrift beziehen würde.

Denn für Prozentwerte gilt allgemein: für jede Eigenschaft, die Prozentwerte unterstützt, ist auch definiert, worauf sich der Prozentwert bezieht. Bei der Schriftgröße beispielsweise bezieht sich der Prozentwert auf **die Schriftgröße des Elternelements**, er kann sich aber auch **je nach Eigenschaft auf einen ganz anderen Wert** beziehen, z.B. bei width auf die Inhaltsbreite des Elternelements.

„Doch wann bezieht sich ein Wert auf einen geerbten Ausgangswert und wann auf eine Voreinstellung im Browser? Die Antwort: Es kommt darauf an.“

Ja, sehr sogar! Die darauf folgende Antwort halte ich für äußerst verwirrend. Die Frage selbst ist sehr unglücklich gewählt, denn was genau ist der Unterschied zwischen „geerbter Ausgangswert“ und „einer Voreinstellung im Browser“? Hinzu kommt, dass es auch Elemente geben kann, bei denen sich eine relative Angabe weder auf einen geerbten Wert, noch auf einen Voreinstellungswert bezieht, sondern auf einen zugewiesenen, den man ja über die Schriftgröße nur zu definieren braucht, doch davon ist keine Rede. Ich vermute allerdings, dass sicher der Author im kompletten Abschnitt „Relative Werte“ unverständlicherweise auf die Eigenschaft font-size bezieht. Auch daher gelingt ihm an dieser Stelle keine ausreichende Erklärung. Und dann schiebt er verwirrender Weise noch die Problematik ein, die Benutzervoreinstellungen und relative Maßeinheiten mit sich bringen können. Daraufhin klärt er die eigentliche Frage nur oberflächlich und zusammenhanglos erst im dritten Absatz auf Seite 62 und im letzten Absatz auf der Seite dann ebenso zusammenhanglos die der Vererbung. Erst auf Seite 157 bekommt man dann eine zusammenhängendere Antwort.

Dennoch möchte ich versuchen, die Frage schon an dieser Stelle zu klären:

Die Voreinstellungen im Browser stellen **die Ausgangswerte** für die Darstellung eines Dokuments dar, die das Wurzelement html (bzw. body) erhält und an Kindelemente weitervererbt. Hat der Benutzer diese Voreinstellungen geändert, so erhält das Wurzelement natürlich diese **geänderten Ausgangswerte**.

Eine em-Angabe bezieht sich immer auf den **font-size-Wert** (ex auf die x-Höhe der Schrift) eines Elements, es sei denn, es wird die Schriftgröße (font-size) eines Elements definiert, denn dann bezieht sich eine em-Angabe auf die Schriftgröße (bei ex auf die x-Höhe der Schrift) des Elternelements, also auf den geerbten Wert.

Das folgende Beispiel macht es deutlicher:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>
<head>
<title>Die relative Schriftgröße</title>
</head>

<body>
<p style="margin:1em;background-color:yellow;">Die relative Schriftgröße </p>

<h1 style="margin:1em;background-color:yellow;">Die relative Schriftgröße </h1>

<h1 style="margin:1em;background-color:yellow; font-size:0.5em">Die relative <span
style="font-size:2em">Schriftgröße</span></h1>

</body>
</html>
```

- Das html- bzw. body-Element erhält die vom Browser bzw. Benutzer **voreingestellten Ausgangswert** für die Schriftgröße. Vorausgesetzt, dass der Benutzer für den p-Elementtyp keine spezifische Schriftgröße voreingestellt hat, erbt es die Schriftgröße von body, die in dem Fall dem **voreingestellten Ausgangswert** des Browsers bzw. Benutzers entspricht.
- Die Schriftgröße des ersten h1-Elements entspricht der Voreinstellung des Browsers für h1-Elemente, z.B. 24pt.
- Die Schriftgröße des zweiten h1-Elements entspricht 0.5em der geerbten Schriftgröße, die schon bei dem p-Element sichtbar ist. font-size bezieht sich hier auf das Elternelement, in dem Fall body, das den Standardwert des Browsers erhalten hat.
- Die Schriftgröße des span-Elements entspricht 2em des zweiten h1-Elements und ist entspricht so insgesamt der body-Schriftgröße (0.5em x 2em = 1em).

Werden relative Wertangaben mit z.B. em oder ex für das Wurzelement (html bzw. body) gemacht, so beziehen sie sich auf den Ausgangswert der Schriftgröße bzw. der x-Höhe der Schrift, also den Voreinstellungen, des Browsers.

Es hängt also von der zu definierenden Eigenschaft und je nach dem entweder der Stellung eines Elements im Dokumentbaum oder von seiner ausdrücklich definierten Schriftgröße ab, welcher konkrete Wert herangezogen wird.

Auf Seite 10 dieser Arbeit mit der Unterüberschrift „[S. 139 Mitte \(dann auch S. 140 Mitte\)](#)“ werden weitere Details rund um die Benutzereinstellungen bzw. Kaskadierung erläutert.

Ich glaube, der Author hätte besser die Frage stellen sollen: „Wie können Browser- und Benutzereinstellungen Stilregeln mit relativen Einheiten beeinflussen?“.

S. 63 Unten

„Zuerst werden wir ein traditionelles, tabellenbasiertes Layout für eine Website erstellen.“

Korrekturen, Erläuterungen und Ergänzungen zu „*Cascading Stylesheets* von Dan Shafer und Kevin Yank

Leider ein leeres Versprechen, denn bis zum Ende wird dieses "tabellenbasierte Layout" nicht erstellt und auch nicht mehr erwähnt.

S. 80

„Beide Seitenbereiche enthalten somit acht Seitenbereiche, die unabhängig voneinander verändert werden können.“

Die Unterseite enthält nur 7 Bereiche, es fehlt nämlich auch der News-Bereich.

S. 82 Mitte

„Im Zentrum einer CSS-Box befindet sich der Inhalt, der mit einem HTML-Element dargestellt wird.“ Verwirrende Aussage, bei der der Eindruck entsteht, der Inhalt einer CSS-Box würde mit einem (zusätzlichen) Element dargestellt werden. Boxen werden für alle Elemente des Dokumentenbaums erzeugt. Die meisten Elemente haben einen Inhalt, der somit Teil der Box ist.

S. 84 Mitte

„In einer CSS-Regel gibt es vier spezifische Eigenschaften, die zur Bestimmung des Innenrands definiert werden können...“. Gemeint ist wohl, dass es in CSS vier Eigenschaften zur Definition von Innenrändern gibt.

S. 85 Mitte und Unten

Diesen Teil halte ich wieder für ziemlich verwirrend, auch wenn es dem Autor daraufhin gelingt, die zunächst gestiftete Verwirrung teilweise zu entwirren. Die folgende Darstellung mag vielleicht gut gemeint sein, macht die Dinge aber unnötig kompliziert und ist auch noch falsch:

„Wie Sie sehen, hat der Innenrand auf der rechten Seite seinen Wert anscheinend nicht angenommen. Obwohl wir...zieht sich die Box weiter bis zum rechten Rand des Browserfensters.“

Ja natürlich, es wurde ja auch kein Außenrand definiert, sondern ein Innenrand. Es geht hier ja um den Abstand zwischen dem Text und der sichtbaren Grenze des Inhaltsbereichs.

„Dieser vermeintliche Fehler liegt nicht an unserer Stilregel, sondern daran, dass h1 ein Blockelement ist und sich damit automatisch über die ganze Seitenbreite erstreckt. Unser definierter Innenrand tritt nur in Kraft, wenn die Breite des Browserfensters kleiner ist, als die Breite der Box des h1-Elements.“

1. Der definierte Innenrand tritt in jedem Fall in Kraft, ist aber nicht in jedem Fall unmittelbar sichtbar. Dass ein rechter Innenrand definiert ist sieht man auch an „gezwungen“ wirkenden Zeilenumbrüchen.
2. Es ist richtig, dass h1 ein Blockelement ist und dass sich Blockelemente automatisch über die gesamte verfügbare Seitenbreite erstreckt. Diese Breite bzw. Anzeigefläche wird aber von der Größe des Browser-Fensters bestimmt. Schleierhaft ist bei der obigen Aussage aus dem Buch, wie dann die Breite des Browserfenster kleiner werden kann als die Breite der h1-Box, wo sich ihre Breite doch automatisch an die des Browserfensters anpasst (steht ja sogar auf Seite 90 unten und 96 oben auch ansatzweise richtig). Eine Ausnahme gibt es da natürlich: wenn die Breite eines Elements mit „width“ vordefiniert wird, kann es passieren, dass die Breite des Browserfensters

kleiner wird als die (des mit einer festen Breite) definierten Elements. Hier ist es aber gewiss nicht der Fall.

Verändert man die Größe des Browserfensters, verändert sich auch die Größe der verfügbaren Anzeigefläche und somit auch die des Blockelements und dessen Inhalt.

Der Inhalt eines Blockelements besteht in der Regel aus Fließtext. Fließtext passt sich stets automatisch der zur Verfügung stehenden Inhaltsbreite eines Blockelements an.

Daher wird der rechte Innenrand auch nur dann sichtbar, wenn der Text bzw. ein Text-Teil aufgrund der verfügbaren Anzeigefläche bis zum rechten Innenrand fließen kann bzw. an ihm umbrochen wird. Der linke ist immer sichtbar, da Text in unserem Sprachraum immer links beginnt, also direkt am definierten Innenrand.

Zusammengefasst:

Ist für ein Block-Element, z.B. h1, ein rechter Innenrand definiert, so ist dieser Rand nur unmittelbar sichtbar, wenn die Inhaltsbreite des Elements aufgrund der Anzeigenbreite des Fensters (body) entsprechenden Raum zur Verfügung hat. Denn dann kann der Inhalt (Text etc.) bis zum definierten Innenrand fließen oder wird je nach Wortlänge an ihm umbrochen und somit deutlich sichtbar.

Auf Seite 88 oben wird dann richtig beschrieben, dass es sich bei der padding-Angabe um einen „Mindestabstand“ handelt.

S. 88 Oben

„Dieser wird eingehalten, wenn ein anderes HTML-Element neben unsere Box positioniert wird oder das Browserfenster kleiner gezogen wird.“

Ich weiß nicht, was hier gemeint ist. Was hat der Innenrand mit der Positionierung von „anderen HTML-Elementen“ zu tun? Der Innenrand wird in jedem Fall eingehalten.

S. 88 Mitte (und auch S. 90 Unten)

„Eine Prozentangabe wiederum stellt den Abstand im Verhältnis zur gesamten Breite des Inhalts in der Box dar.“

Diese Aussage ist in zweifacher Hinsicht falsch. Weder bezieht sich die Prozentangabe auf die gesamte Inhaltsbreite, noch bezieht sie sich auf das Element, für das sie angegeben wird.

Sie bezieht sich auf **die gesamte Box-Breite des Eltern-Elements**. In diesem Fall ist body das Eltern-Element. Daher bezieht sich die Prozentangabe in diesem Fall also auf die verfügbare Anzeigefläche des Browsers.

Bitte auch das Folgende lesen:

S. 91 Unten

„Der Unterschied der beiden Ränder eines Elements liegt darin, dass mit dem Außenrand immer eine Fläche außerhalb der Box definiert wird, während der Innenrand eine Fläche innerhalb der Box darstellt.“

Präziser wäre diese Aussage, wenn es hieße: „...dass mit dem Außenrand immer eine Fläche außerhalb des sichtbaren Bereichs einer Box (dazu zähle ich

Inhaltskasten, Polsterung und Rahmen) definiert wird. Denn das CSS-Box-Modell beschreibt eine umfassende Box, also einen äußeren Kasten, der für Elemente erstellt wird und wiederum selber innere Kästen (Rand, Rahmen, Polsterung und Inhalt) umschließt. Mit margin wird der Randbereich, also der Bereich zwischen dem Rahmen (border) und der Randkante der Box definiert. Wesentlich ist, dass **auch er zur Box gehört**.

S. 92 Unten

„Um die Auswirkungen der beiden Ränder noch deutlicher gegenüberzustellen, weisen wir einem der HTML-Elemente einen Innenrand und dem anderen einen Außenrand zu.“

Es wird aber beiden HTML-Elementen ein Innenrand zugewiesen.

S. 95 Unten

„Die letzte Stilregel im Quelltext weist einem p, das einem li-Element direkt untergeordnet ist...“

Die Aussage ist nicht ganz korrekt, denn es handelt sich um einen Selektor für Nachfahren und nicht für Kindelemente. Deshalb muss das p-Element dem li nicht direkt untergeordnet sein, sondern kann beispielsweise in einem div-Element innerhalb des li-Elements verschachtelt sein, also li div p.

S. 99 Mitte

„Der Rahmen, der Innen- vom Außenrand voneinander trennt...“

„Innen- **und** Außenrand voneinander trennt“ macht weniger stutzig...

S.101 Unten

„Die Voreinstellung für die display-Eigenschaft richtet sich nach dem Elementtyp, dem sie zugewiesen wird.“

Der Nebensatz ist wohl überflüssig.

S. 102 Mitte

„Dafür werden die einzelnen HTML-Elemente zu einem div-Bereich zusammengefasst und...“

HTML-Elemente können **in** einem div-Bereich zusammengefasst werden.

S. 102 Unten

„...jedes Element wird immer relativ zu seinem übergeordneten Element positioniert.“

Diese Aussage ist falsch, wie sich auch beim Weiterlesen herausstellt. Es gibt zwei Möglichkeiten der Positionierung mit der CSS-Eigenschaft position: absolut und relativ.

Nur bei „**absolut**“ wird relativ zum übergeordneten Element positioniert, bei „**relative**“ relativ zur normalen (Fließtext-)Position des Elements selbst im Dokument.

S. 104 Mitte

„Ihre Wirkung auf den zweiten div-Bereich multipliziert sich, da dieser die Regel einmal als untergeordnetes Element des ersten div-Bereichs vererbt bekommt...“

Regeln werden nicht vererbt. Hier wird nur die berechnete Schriftgröße vererbt.

S. 105 Unten

„Die absolute Positionierung des Elements wird immer am übergeordneten Element ausgerichtet.“

Wird das Element ausgerichtet oder seine Positionierung? Abgesehen davon ist die Aussage mehr als unpräzise bzw. falsch. Ein Element wird bei der absoluten Positionierung nur dann an seinem übergeordneten Element ausgerichtet, wenn dieses (das übergeordnete Element) **für position einen anderen Wert als static hat**. Ansonsten wird das Element „einfach“ gesagt an body ausgerichtet. Siehe auch S. 380 in der Referenz.

S. 107 Unten

Die Fußnote enthält einen Rechenfehler. Oben auf der Seite wurde noch richtig gerechnet. Unten ergibt dann 220px-200px auf einmal 10px.

Dabei sind es ja $220\text{px} - (200\text{px} + 10\text{px Au\ss enrand}) = 10\text{px}$

S. 112 Oben

„Wenn man mehrere Elemente nebeneinander anordnet, muss die Summe ihrer Breitenwerte der Gesamtbreite des Blocks entsprechen, in dem sie sich befinden.“

Diese Aussage ist natürlich richtig. Allerdings steht dann weiter unten, dass als Breite die width-Eigenschaft im Box-Modell, also die Inhaltsbreite gemeint ist.

Haben aber Elemente, die nebeneinander angeordnet werden sollen, beispielsweise noch zusätzlich einen Innenrand oder einen Rahmen, so muss dieser selbstverständlich auch berücksichtigt werden.

S. 112 Mitte

„Er schließt Rahmen, Außen- und Innenränder fälschlicherweise bei width und height mit ein.“

Der Internet Explorer schließt die Außenränder nicht mit ein.

S. 112 Unten

„Leider hilft uns das nicht weiter, wenn wir Elemente auf der gesamten Darstellungsfläche einer Seite definieren, da diese sich in ihrer Breite immer an die Größe des Browserfensters anpasst.“

Diese Aussage mutet merkwürdig. Es könnte gemeint sein: „...wenn wir Elemente über die gesamte Darstellungsfläche einer Seite definieren...“

Außerdem kann man auch mit relativen Angaben über die gesamte, flexible Darstellungsfläche ein konsistentes Layout aufbauen, das zwar keine fixe Breite hat, dafür aber fixe Proportionen, wie der Author später ja selber zeigt.

S. 113 Fußnote

„Wir könnten die hängende Initiale ebenso mit :first-letter erstellen.“

Es besteht ein stilistischer Unterschied zwischen einer hängenden Initiale mit float in Verbindung mit span (Abb. rechts erster Absatz in blau) und :first-letter (Abb. rechts zweiter Absatz in schwarz).

Übrigens: Ob die hängende Initiale mit float zu dem ihr folgenden Text bündig erscheint ist im Gegensatz zu :first-letter nicht gesagt, denn sie darf nicht allzu groß sein.



S. 115 Oben

Mit „float-Box“ ist hier das mit float positionierte Element gemeint.

Korrekturen, Erläuterungen und Ergänzungen zu „*Cascading Stylesheets* von Dan Shafer und Kevin Yank

S. 119 Unten

Falls Sie sich fragen sollten: Ja, die Quelltexte 6-6 und 6-7 (von der dpunkt-Seite zum Buch) sind identisch. 6-7 hätte man sich wirklich sparen können.

S. 120 Mitte (grauer Kasten)

„Wenn Sie ein Element mit einem Abstand zum Rand der Seite positionieren möchten, sollten Sie immer zuerst den Außenrand der ganzen Seite auf null setzen.“ Diesen Rat sollte man beherzigen. Es ist allerdings auch gut zu wissen, dass die Außenränder von body für absolut-positionierte (z.B. mit top und left) Kindelemente keine Rolle spielen. Was heißt das? Dieses Beispiel macht es deutlich:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Positionierung</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<style type="text/css">
<!--
body {
margin:150px; background-color:yellow;
}
-->
</style>
</head>
<body>
<p style="position:absolute;top:10px;left:50px;background-color:white;">Dieser Absatz wurde
absolut positioniert und befindet sich 10 Pixel unterhalb und 50 Pixel rechts von der oberen,
rechten Ecke der Anzeigefläche</p>
<p>Ein einfacher Absatz</p>
</body>
</html>
```

Obwohl body einen Außenrand von jeweils 150px hat, wird der Absatz dennoch 10px vom oberen Anzeigen- und 50 Pixel vom linken Anzeigenrand positioniert.

S. 121 Oben

„Diesen Fehler können wir vermeiden, indem wir die linke und rechte Spalte mit absoluten Werten definieren und...“

Die rechte Spalte wird offensichtlich nicht mit absoluten Werten definiert, sondern mit relativen.

Die mittlere Spalte in Abbildung 6-8 hat keine 60% Breite, sondern 60%-100px, was insbesondere bei kleinen Fenstergrößen auffällt.

S. 124 Kasten

Wenn sich statisch positionierte Blockelemente tatsächlich an dem oberen Außenrand ihres ersten untergeordneten Elements ausrichten sollen, dann wäre die folgende Stilregel vermutlich besser: `div:firstchild {margin-top:0;}`. Die Pseudoklasse `:firstchild` wird aber leider vom IE nicht unterstützt.

S. 132 Unten

„Diese Stilregel setzt den rechten und linken Außenrand des Bildbereichs auf 25%.“ Gemeint ist nicht der Bild-, sondern der Inhaltsbereich.

S. 133 Unten Quelltext

Korrekturen, Erläuterungen und Ergänzungen zu *„Cascading Stylesheets“* von Dan Shafer und Kevin Yank

Wichtiger Hinweis:

Für den Bereich #left (auch für #sponsor und #news) fehlt die CSS-Eigenschaft left:0;. Sie sollte ergänzt werden, um insbesondere Komplikationen mit dem Internet Explorer zu vermeiden. Die Website funktioniert zwar auch fehlerfrei ohne diese Angabe, doch das liegt an der Anordnung der Elemente im HTML-Quelltext. Befände sich beispielsweise der #left-div-Bereich direkt vor dem #center-div-Bereich im HTML-Quelltext, so würde das Layout im IE nicht mehr passen (genauso auch für die anderen beiden genannten), es muss also auf die Reihenfolge der definierten Bereiche geachtet werden. Mit der zusätzlichen Angabe left: 0; spielt die Reihenfolge keine Rolle.

S. 134 Unten

„Mit 75% Außen- und zweimal 1,5% Innenrand ergeben sich insgesamt 78%, die für die Spalte übrig bleiben.“

Also, nach meiner Rechnung bleiben für die Spalte 22% übrig.

S. 139 Mitte (dann auch S. 140 Mitte)

„Nach dem CSS2-Standard überschreiben die lokalen Voreinstellungen des Benutzers im Browser grundsätzlich die Stilregeln und HTML-Attribute, die eine Website mit sich bringt.“

Diese Aussage ist im Zusammenhang mit dem dann folgenden Text nicht ganz richtig, zumindest aber unvollständig und irreführend. Denn „die lokalen Voreinstellungen“ des Benutzers können, müssen aber nicht „grundsätzlich“ die Stilregeln und HTML-Attribute einer Website überschreiben. Das kommt wirklich darauf an. Dazu muss ich ein wenig ausholen:

Nach den CSS2-Kaskadierungsregeln der **Herkunft und Gewichtung** (s. auch 202 im Buch) können Stylesheets drei verschiedene Ursprünge haben:

- Autor (also z.B. vom Designer einer Seite)
- Benutzer (der die Seite mit einem Browsers anzeigt und eben seinen visuellen Wünschen und Bedürfnissen gemäß auch eigene Stylesheets definieren kann)
- Benutzerprogramm (Standardeinstellungen bzw. Werte des Browsers selbst).

Dabei gilt standardgemäß folgende Reihenfolge bei der Umsetzung:

Die Autoren-Stylesheets haben im Konfliktfall mehr Gewicht als die Benutzer-Stylesheets und diese wiederum mehr als die des Benutzerprogramms.

Das ist die „natürliche“ Rangfolge bei der Anwendung von Stylesheets und so reagiert beispielsweise auch der Internet Explorer 6 für Windows und Opera 7.22: die Autoren-Stylesheets haben standardmäßig, falls nichts Spezielleres eingestellt, Vorrang vor dem ausgewählten Benutzer-Stylesheet.

Nur wenn das Benutzer-Stylesheet eine **!important-Deklaration** (s. auch S. 202-209 im Buch) enthält, gilt diese laut CSS2 in jedem Fall vor einer entsprechenden Deklaration im Autoren-Stylesheet, selbst dann, wenn eine entsprechende Regel im Autoren-Stylesheet mit !important definiert werden würde.

Ein Beispiel zum besseren Verständnis:

```
/*Benutzer-Stylesheet */
p {color: red;}
p {font-size: 18pt !important}
```

```
/*Autor-Stylesheet*/  
p {color: blue;}  
p {font-size: 12pt !important}
```

Bei dem Beispiel handelt es sich um zwei Stylesheets mit unterschiedlicher Herkunft: Autor und Benutzer. Also Folge erhalten die Absätze eine **blaue Textfarbe**, da standardmäßig die Deklarationen des Autors die des Benutzers überschreiben. Doch sie erhalten eine Schriftgröße von 18 Punkt, da die entsprechende Deklaration des Benutzers durch die Anweisung !important mehr Gewicht hat als die des Autors, obwohl auch diese mit !important gekennzeichnet ist. Der Internet Explorer 6 reagiert hier wie beschrieben standardkonform, Opera 7.22 leider nicht: er gibt einer mit !important gekennzeichneten Benutzer-Deklaration trotzdem nicht den Vorrang.

Zugegeben, das Thema ist nicht ganz einfach und am Besten man experimentiert ein wenig.

Alles in allem ist es wichtig festzuhalten:

Natürlich ist es nicht ausgeschlossen, dass Benutzer-Stylesheets Autoren-Stylesheets überschreiben und das ganz ohne !important-Regeln. Das liegt einfach daran, dass sich die meisten Browser (z.B. Internet Explorer 6 oder Opera 7.0) auch so konfigurieren lassen, dass lokale Stylesheets (also Standard- und Benutzer-Stylesheets) in jedem Fall Vorrang vor Autoren-Stylesheets haben und/oder diese erst gar nicht geladen werden. Das Anzeigen von Bildern lässt sich ja z.B. auch unterdrücken.

Wie im Buch richtig beschrieben wird, hängt es eben stark von den Einstellungen des Benutzers ab, welche Stylesheets zum Zuge kommen: die vom Autor, die vom Benutzer, beide (wobei dann vorrangige Deklarationen gelten) oder sogar keine. Allerdings wird im Buch an der Stelle unverständlicherweise vom Internet Explorer 5 für Macintosh ausgegangen und es entsteht der Eindruck, dass falls der Benutzer ein lokales Stylesheet erstellt und verwendet, dieses in jedem Fall die Autoren-Stylesheets überschreibt. Es heißt auf Seite 140 Mitte: *„Sogar ein eigenes lokales Stylesheet kann im Browser erstellt werden, das automatisch für jede Seite zum Einsatz kommt. Definieren wir darin zum Beispiel für das Element h1 die Farbe Blau, wird jede Überschrift erster Ordnung so dargestellt, ganz gleich, unter welchen Vorgaben ihr Webdesigner sie angelegt hat.“* Nebenläufig wird dann auf Seite 147 in einer Fußnote erwähnt, dass das z.B. im Internet Explorer 6 nur auf ausdrückliche Anweisung des Benutzers geschieht.

Es war mir wichtig genau aufzuzeigen, dass es laut CSS2-Spezifikation nicht das Standard-Verhalten eines Browsers ist, Autoren-Stylesheets mit lokalen Stylesheets zu überschreiben, er aber sehr wohl dementsprechend eingestellt sein kann.

Der Merksatz aus dem Buch „Gilt nur, wenn nicht anderes in den Benutzereinstellungen festgelegt!“ passt gut. Es ist aber auch gut zu wissen, welches Browser-Standard-Verhalten die CSS2-Spezifikation im Konfliktfall von Regeln (Kaskadierung) vorgesehen hat.

Übrigens gilt für die Anwendung von HTML-Elementen und Attributen mit Gestaltungsfunktion (z.B. font oder bgcolor):

Allgemein überschreiben Stylesheets im Konfliktfall entsprechende HTML-Elemente bzw. Attribute. Außerdem haben HTML-Elemente bzw. Attribute mit Gestaltungsfunktion im Dokument normalerweise Vorrang vor den entsprechenden Darstellungseinstellungen des Browsers (allerdings nicht vor lokalen Stylesheets!), es sei denn, der Benutzer

konfiguriert seinen Browser ausdrücklich so, dass er seinen eigenen Darstellungseinstellungen den Vorrang gibt.

S. 143 Fußnote

Die unten angegebene Webseite hat eine neue Adresse:

<http://r0k.us/graphics/SlHwheel.html>

S. 144 Fußnote

Die unten angegebene Webseite gibt es nicht mehr.

S. 146 Mitte und Unten

„Mit einem Trick können wir für jedes Element einzeln sicherstellen, dass seine Hintergrundfarbe identisch mit der des body-Elements ist.“

So beginnt der kurze, aber wichtige Abschnitt **„7.6 Transparenz, Farbe und Benutzereinstellungen“**. Er ist leider stellenweise gravierend falsch:

Hier wird eine Klasse *transbox* angelegt, in der die Hintergrundfarbe als transparent definiert wird. Dann heißt es in der Mitte der Seite (mit der Randbemerkung *„Hintergrundbilder“*), dass die Eigenschaft `background-color` standardmäßig immer auf transparent gesetzt und vom Vater-Element nicht vererbt werden würde.

Daraufhin heißt es (für Hintergrundfarben) im Widerspruch zur vorherigen Aussage *„Wenn jedoch ein Element einen farbigen Hintergrund zugewiesen bekommt, vererbt es diese Eigenschaft an alle untergeordneten Elemente weiter. Möchten wir das für ein bestimmtes Element verhindern, weisen wir es einfach der Klasse unserer Stilregel *transbox* zu. Der vererbte Wert wird überschrieben und der Hintergrund ist wieder durchsichtig bis zur body Einstellung.“*

Wie zunächst richtig geschrieben wird, hat die Eigenschaft **background-color für alle Elemente den Standardwert transparent**. Es ist auch richtig, dass eine Hintergrundgrafik nicht vererbt wird. Das gilt allerdings auch für eine Hintergrundfarbe, was im Buch ausdrücklich falsch steht. Der Grund, warum die Hintergrundgrafik und -farbe von Elementen gleich der ihrer Eltern ist, liegt ja gerade daran, dass alle Elemente als Ausgangswert für `background-color` transparent haben. Somit kann die Hintergrundfarbe des Elternelements durchscheinen. Alles andere ist einfach nur Unsinn, **weder Hintergrundgrafiken noch Hintergrundfarben werden vererbt**. Hier wird mit der definierten Klasse dem Wind das Wehen beigebracht.

Rein hypothetisch:

Selbst wenn sie vererbt werden würden, was ja **nicht** der Fall ist, könnte der Hintergrund nicht in jedem Fall durchsichtig bis zur body-Einstellung sein. In Annäherung an das Prinzip der Stapelung müsste beispielsweise eine Überschrift `h1` mit der Klasse *transbox* als Kindelement eines `div`-Bereichs mit der Hintergrundfarbe von z.B. gelb die Hintergrundfarbe seines Elternelements (gelb) zeigen und nicht die von `body` (mit irgendeiner anderen Farbe), da ja die Überschrift innerhalb der `div`-Box existiert. Siehe auch Quelltext unten.

Probieren Sie die vom Autor beschriebene „Technik“ der *transbox* bitte selbst aus, um sich bewusst zu machen, warum sie nicht funktionieren kann. Hier ein entsprechender Quelltext:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <title>Transbox</title>
<style type="text/css">
```

Korrekturen, Erläuterungen und Ergänzungen zu *„Cascading Stylesheets von Dan Shafer und Kevin Yank*

```

<!--
.transbox {
color:white;
background-color:transparent;
}
-->
</style>
</head>
<body style="background-color:blue">
<h1>Bin ich durchsichtig? Ja klar bin ich das. Das ist meine Standard-Einstellung.</h1>
<h1 class="transbox">Bin ich durchsichtig? Ja klar, war ich auch schon vor der
Klassenzuweisung.</h1>

<div style="background-color:purple;">
<h1>Bin ich durchsichtig? Ja klar...</h1>
<h1 class="transbox">Bin ich durchsichtig? Ja natürlich bin ich das, war ich auch schon vor
der Klassenzuweisung. Wie weit können Sie durch mich hindurch schauen?</h1>
</div>

</body>
</html>

```

Übrigens: Im Referenzteil des Buchs unter background-color und background-image (S. 282 und S. 283) steht das dann auch alles korrekt drin.

S. 160 Mitte

„Zur Verfügung stehen laut CSS-Spezifikation auch zwei Konstanten normal und bold...beide werden jedoch von keinem aktuellen Browser unterstützt.“

Beide Werte werden von allen aktuellen Browsern unterstützt.

S. 163 Oben

„Standardisierte und nicht standardisierte Schrifttypen“

Das ist die Überschrift, die der Autor unverständlicherweise für das Kapitel 8.6 gewählt hat. Dann schreibt er ein paar Zeilen später: *„Schon der Begriff Standard ist bei Schriften etwas irreführend, denn erstens gibt es keinen Standard und keine Spezifikationen, die vorschreibt, welche Schriften im System des Benutzers installiert sein müssen.“*

Wieso wählt er dann so einen irreführenden Begriff als Überschrift? Im weiteren Verlauf des Kapitels wird dann mit keinem Wort mehr darauf eingegangen, was denn eine standardisierte bzw. nicht standardisierte Schrift sein soll und plötzlich ist von alternativen Schriften die Rede. Mit Standardisierter Schrift ist wohl einfach eine weit verbreitete Schrift gemeint, die auf vielen Rechnern zu finden ist. Auf Seite 166 kann man dann auch aus der Überschrift „Nicht standardisierte [...] Schriften“ schließen, dass damit eine seltene, auf wenigen Rechnern vorhandene Schrift gemeint sein soll.

S. 179 Unten

„Beachten Sie, dass sich relative Werte in line-height anders verhalten als bei den meisten anderen Schrifteigenschaften: Relative Werte in line-height benutzen als Ausgangswert immer die Schriftgröße jenes Elements, für das sie definiert werden, und nicht etwa die Werte ihre übergeordneten Elements (wie zum Beispiel bei font-size).“

Eigentlich ist es ja umgekehrt: nur bei der Verwendung von relativen Werten zur Festlegung der Schriftgröße(font-size) eines Elements bezieht sich ein relativer Wert auf den Schriftgrößen-Wert des Elternelements. Bei der Verwendung von relativen Werten mit allen anderen Schrifteigenschaften bezieht sich der Wert auf die Schriftgröße des Elements selbst.

S. 181 Unten

Korrekturen, Erläuterungen und Ergänzungen zu „*Cascading Stylesheets* von Dan Shafer und Kevin Yank

„Die für den div-Bereich gültige und an den p-Absatz vererbte Zeilenhöhe errechnet sich als die doppelte Schriftgröße des div-Elements.“

Offensichtlich berechnet sich die Schriftgröße als die 1,5-fache Schriftgröße und nicht als die doppelte.

S. 189 Oben

„Die Rahmenlinie über der zweiten Überschrift verläuft über die ganze Seitenbreite, weil sie das Kopfende der Box des Blockelements h1 anzeigt.“

Eigentlich zeigt die Rahmenlinie nicht das Kopfende der Box von h1 an. Das Kopfende stellt die obere margin-Kante des margin-Kastens dar. In der Zeichnung auf der Seite 82 im Buch kann man das auch gut erkennen.

S. 192 Mitte

„Zur Erinnerung: Mit der Angabe margin:0px im div-Bereich umgehen wir den Mozilla Fehler am oberen Seitenrand, der bereits in Kapitel 6 ausführlich beschrieben wurde.“

In Kapitel 6 auf der Seite 124 ist von statisch positionierten Elementen und ihr erstes untergeordnetes Element die Rede. In dem Beispiel auf Seite 191 ist aber div nicht statisch positioniert und es erhält auch nicht das erste untergeordnete Element margin:0px, sondern das zweite. Genaueres zum „Mozilla-Bug“ weiß ich nicht.

S. 193 Unten

„Nur seine Unterstreichung ist im Browser ausgeschaltet worden.“

Das ist offenbar nicht der Fall.

S. 194 Mitte

Im Quelltext wird für die Eigenschaft background-color der Wert „transparent“ definiert, die aber standardmäßig schon diesen Wert hat.

S. 195 Mitte

„Beide sind umschließende Tags und benötigen die Abschluss-Tags oder am Ende ihre Elements.“

Eine merkwürdige und verwirrende Formulierung. Tags haben keine Elemente. Elemente haben Tags die ihren Anfang und ihr Ende kennzeichnen.

Besser müsste es so heißen: „Beide sind umschließende Block-Elemente und benötigen die Abschluss-Tags.“

S. 197 Mitte

„Für verschachtelte Listen wird auf der ersten Einzugsebene ein ausgefülltes Quadrat eingesetzt.“

Das ist offensichtlich nicht der Fall, denn wie auf dem Bild auf der Seite 197 zu sehen ist es kein Quadrat sondern ein Kreis.

S. 198 Oben

„Bitte beachten Sie, dass wir hier für die drei verschachtelten Listenebenen und ihre zugehörigen Stilmerkmale Selektoren für Nachfahren einsetzen.“

Es sind zwei verschachtelte Listenebenen bzw. nur für diese Selektoren für Nachfahren.

S. 202 Unten

Korrekturen, Erläuterungen und Ergänzungen zu „*Cascading Stylesheets* von Dan Shafer und Kevin Yank

„Die Kaskadierung bestimmt, welche Eigenschaften zum Einsatz kommen, wenn ein Element Werte von mehreren Stilregeln zugewiesen bekommt.“

Es geht nicht um Eigenschaften, sondern um Deklarationen bzw. um Eigenschaftswerte, die zum Einsatz kommen. Angewendet werden ja alle (korrekt) angegebenen Eigenschaften. Besser wäre also die Formulierung: „...welche Deklarationen zum Einsatz kommen, wenn ein Element die gleichen Eigenschaften durch mehrere Stilregeln zugewiesen bekommt.“ Auch auf den folgenden Seiten ist stellenweise von „Eigenschaften“ die Rede, wo besser von Deklarationen gesprochen werden sollte (z.B. S. 206 Oben in der Mitte, S. 208 Unten).

S. 203 Unten

„Der Browser durchsucht die für das Element gültigen Stilregeln nach dem Schlüsselwort !important.“

Eigentlich werden die Deklarationen nach dem Schlüsselwort !important durchsucht.

S. 207 Unten

„Keine andere externe oder eingebettete Stilregel kann eine interne Stileigenschaft überschreiben.“

Gemeint ist natürlich, dass keine andere externe oder eingebettete Stilregel aufgrund ihrer Spezifität eine interne mit dem Attribut style deklarierte überschreiben kann.

S. 227 Mitte

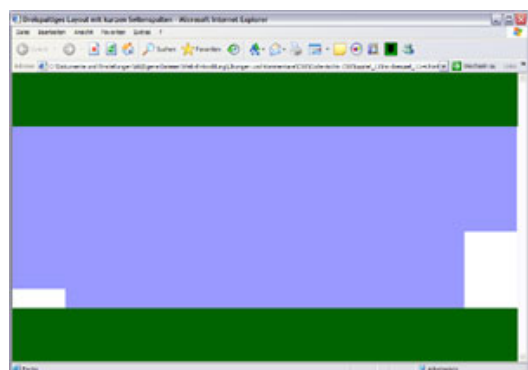
„CSS bietet keine Methode dafür an, sie in derselben Höhe anzuzeigen, es sei denn mit absoluten Höhenwerten. Solche führen jedoch zu weiteren Problemen und sind daher zu vermeiden.“

CSS bietet außerdem mit der Eigenschaft min-height die Möglichkeit an, allen drei Spalten eine Mindesthöhe zuzuweisen. Dieser Ansatz ist natürlich nicht so flexibel, wie die im Buch beschriebene Methode mit JavaScript, aber u. U. eine brauchbare Lösung. Leider wird diese Eigenschaft vom IE nicht unterstützt.

S. 227 Unten

„Der erste der beiden geschilderten Fälle bleibt unsichtbar, wenn man in allen Spalten mit der gleichen Hintergrundfarbe und ohne Rahmen arbeitet.“

Es ist bei unterschiedlicher Spaltenlänge und gleicher Hintergrundfarbe für alle drei Spalten anhand der entstehenden „Lücken-Blöcke“ links und rechts sehr wohl zu sehen, dass die Spalten unterschiedlich lang sind. Nur wenn sie keine Hintergrundfarbe hätten oder die ihres gemeinsamen Elternelements (body), wäre der Fehler unsichtbar.



Übrigens: Beide Mängel haben offensichtlich ein und dieselbe Ursache: sie rühren daher, dass jeweils der Inhalt der drei Spalten maßgeblich ist für die jeweilige Spaltenlänge. Fall zwei ist also einfach nur ein Sonderfall von Fall 1.

S. 228 Mitte

Ein flexibles Layout ist mit diesem Ansatz nicht möglich, da weder der Internet Explorer, noch Mozilla oder Opera %-Angaben für border akzeptieren.

S. 230 Oben

„Weil beide Vorgänge nur für zwei getrennte Elemente ablaufen können, muss jeder div-Bereich einen weiteren div-Bereich enthalten.“

Diese Behauptung war für mich in keiner Weise nachvollziehbar. Es werden im darauf folgenden Script die Höhenwerte der Spalten ermittelt, verglichen und der größte Höhenwert allen Spalten zugewiesen. Wozu die weiteren div-Bereiche? Es funktioniert auch korrekt ohne diese.

Irritierenderweise steht sogar in der Korrektur des Verlages auf Seite 3 „Bei der Zuweisung der neu berechneten Werte wurde aber ein falscher Ort angegeben.“ Doch selbst dieser vermeintlich fehlerhafte Code funktioniert einwandfrei, was einfach an der „neutralen“ Natur von div liegt.

S. 256 Oben

„Schwieriger wird die Transformation von CSS-Eigenschaften in HTML, wenn Selektoren verwendet werden, die sich auf spezifische Elemente beziehen. In diesem Fall bleibt uns nichts anderes übrig, als die Änderungen für jedes einzelne HTML-Element selbst durchzuführen.“

Diese abschließende Aussage ist merkwürdig. Denn auch schon auf Seite 255 unten handelt es sich ja bei dem Beispiel mit den Schrifteigenschaften um einen Selektor (p). Folglich müssen also alle HTML-Elemente, in dem Fall also Absätze, einzeln mit einem font-Element ausgezeichnet werden. Von daher ist es merkwürdig, warum plötzlich im darauf folgenden Abschnitt mit der Randnotiz „Selektoren für spezifische Elemente“ von „schwieriger“ die Rede ist. Es ging ja schon vorher um Selektoren. Die Änderungen müssen doch in jedem Fall für jedes einzelne HTML-Element durchgeführt werden.

S. 259 Fußzeile 6.

Die Website ist nicht mehr verfügbar.

S. 260

„Einsatz von DOCTYPE-Switching“

Das Doctype-Switching wird hauptsächlich eingesetzt, um die fehlerhafte Darstellung von Element-Boxen im Internet Explorer (Box-Model-Bug) zu korrigieren.

S. 267 Unten

„Wenn Sie für einen Medientyp ein Stylesheet definieren, das den Begriff einer Druckseite versteht...“

Korrekt muss es heißen: „...der den Begriff einer Druckseite versteht.“

S. 269 Oben

„Die meisten Anwendungen, die Webinhalte akustisch präsentieren, richten sich bisher nur an hörgeschädigte Menschen.“

Gemeint ist wohl sehgeschädigte Menschen.