Mobile Webseiten & Apps

Mobiles Web

von Kopf bis Fuß

Ein Mal entwickeln, unbegrenzt ausliefern

Zeigen Sie Einsatz (für Ihre User)

Orientieren Sie sich dank Geolocation

Bringen Sie Ihre Sites mit Responsive Webdesign in Form

Setzen Sie Ihre Seiten auf eine Display-Diät

O'REILLY®

Lysar Danger Gardener & Jason Grigsby
Übersetzung von Lars Schulten

1 Einstieg in die mobile Webprogrammierung

Responsive Webdesign

Wie sieht es aus? Sind Sie bereit für den Sprung in die mobile Welt?

Die mobile Webentwicklung schafft ein extrem aufregendes Lebensgefühl. Man findet dort Glanz und Gloria und viele ***Aha!***-Momente. Aber man wird auch mit Mysterien und Verwirrung konfrontiert. Die Mobiltechnologie entwickelt sich in halsbrecherischer Geschwindigkeit und stellt einen permanent vor neue Herausforderungen! Halten Sie sich also fest. Wir beginnen unsere Reise, indem wir Ihnen ein Verfahren zum Aufbau von Websites vorstellen, das als ***Responsive Webdesign (RWD)*** bezeichnet wird. Mit dessen Hilfe werden Sie Websites so anpassen können, dass sie ausgezeichnet auf Mobilgeräten aussehen, und Sie brauchen dazu nur Fertigkeiten, die Sie bereits besitzen.

Springen Sie auf den Mobilzug auf

Es ist ziemlich wahrscheinlich, dass Sie ein Handy besitzen. Das wissen wir nicht nur, weil Sie dieses Buch erworben haben (nebenbei: kluger Schachzug!), sondern auch, weil es nicht gerade einfach ist, jemanden zu finden, der keines besitzt.

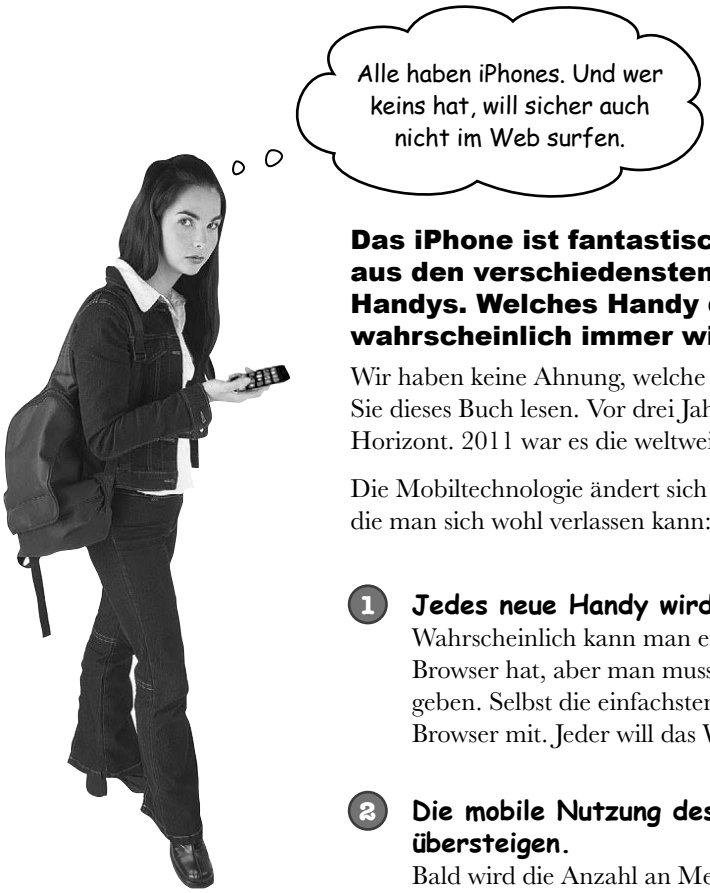
Ganz gleich wohin Sie auf diesem Planeten reisen, Handys werden überall genutzt: Nigerianische Farmer nutzen sie, um herauszufinden, auf welchem Markt sie den besten Preis für ihre Ernte erzielen, und in Japan werden die zehn meistverkauften Bücher darauf gelesen und geschrieben – *Sie haben richtig gehört, geschrieben*.

Anfang 2011 wurden von den 6,9 Milliarden Menschen auf dieser Erde 5,2 Milliarden Handys genutzt. **Es gibt mehr Menschen, die ein Handy besitzen, als Menschen, die funktionierende Toiletten oder Zahnbürsten haben.**

Die Stunde hat geschlagen

Mobile Geräte sind also eine große Sache. Toll, ist das nicht schon seit Jahren so? Warum sollten Sie ausgerechnet jetzt auf den Zug aufspringen?

Weil **das iPhone alles geändert hat**. Das klingt nach einem Klischee und ist dennoch wahr. Auch vor dem iPhone gab es schon App Stores, Touchscreens und Webbrowser für Handys, aber erst Apple verstand es, alles so zusammenzufügen, dass es Otto Normalverbraucher verstehen und nutzen konnte.



Das iPhone ist fantastisch, aber die Menschen nutzen aus den verschiedensten Gründen ganz unterschiedliche Handys. Welches Handy das beliebteste ist, wird sich wahrscheinlich immer wieder ändern.

Wir haben keine Ahnung, welche Handys die wichtigsten sein werden, wenn Sie dieses Buch lesen. Vor drei Jahren war Android nur ein kleiner Punkt am Horizont. 2011 war es die weltweit führende Smartphone-Plattform.

Die Mobiltechnologie ändert sich zwar rasant, aber es gibt einige Punkte, auf die man sich wohl verlassen kann:

1. **Jedes neue Handy wird einen Browser haben.**
   Wahrscheinlich kann man ein neues Handy finden, das keinen Browser hat, aber man muss sich dabei schon eine Menge Mühe geben. Selbst die einfachsten Handys bringen mittlerweile ordentliche Browser mit. Jeder will das Web in der Hosentasche tragen.

2. **Die mobile Nutzung des Webs wird die Desktop-Nutzung übersteigen.**
   Bald wird die Anzahl an Menschen, die über Mobilgeräte auf das Web zugreifen, die Anzahl derjenigen übersteigen, die das mit einem Computer tun. Schon heute geben viele an, ihr Handy häufiger zu nutzen als ihren PC.

3. **Das Web ist die einzige wahre plattformübergreifende Technologie.**
   iPhone, Android, BlackBerry, Windows Phone, WebOS, Symbian, Bada – es gibt mehr Smartphone-Plattformen, als man sich merken kann. Jede hat ihre eigene Programmierschnittstelle: Wenn Sie Software für diese unterschiedlichen Plattformen schreiben wollen, müssen Sie also jedes Mal bei null beginnen.

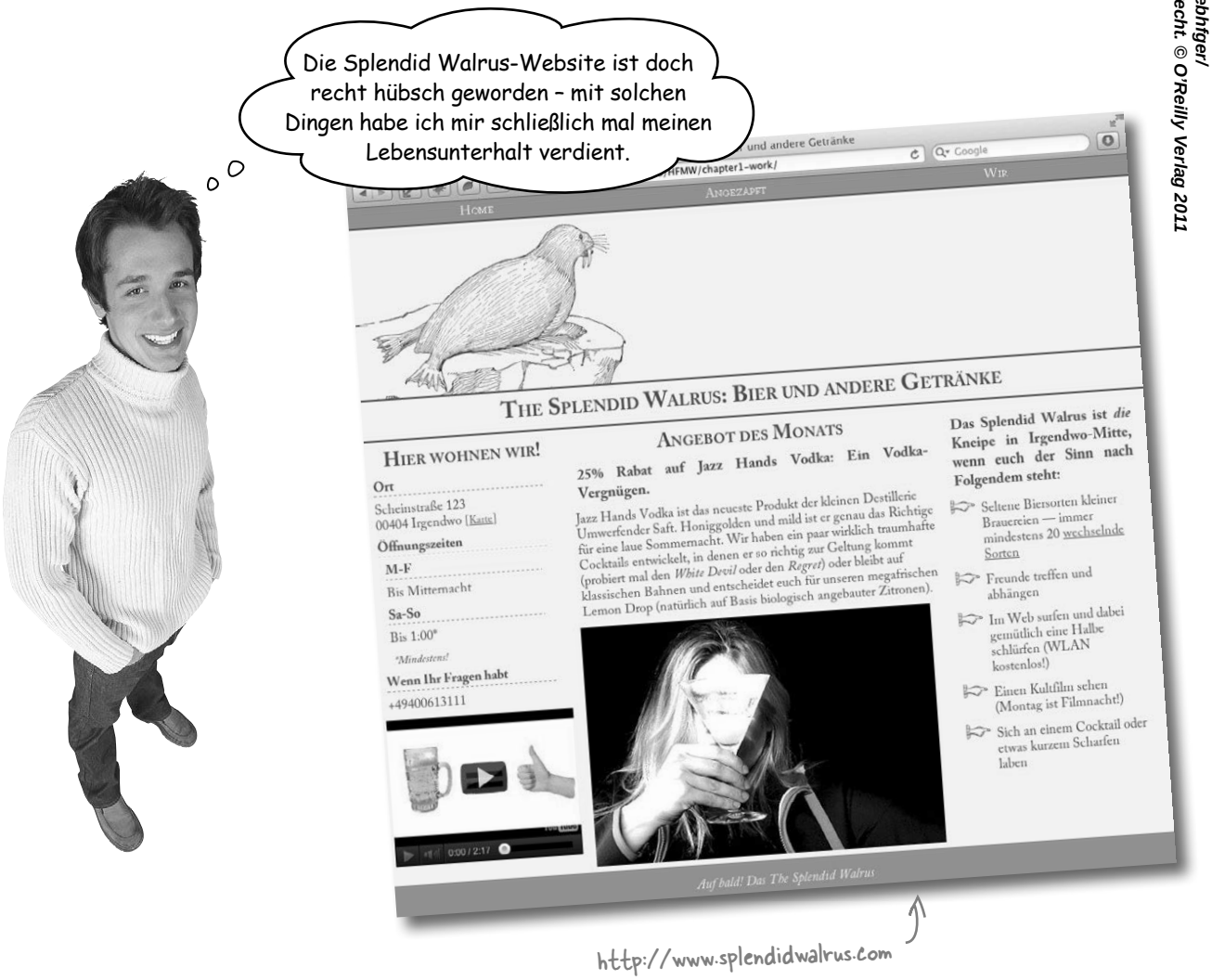
Das mobile Web bietet eigene Herausforderungen, aber es gibt dennoch keine andere Technologie, mit der Sie Inhalte und Apps erstellen können, die alle Plattformen erreichen.

Sie sind also zur rechten Zeit am rechten Ort. Das mobile Web hebt ab, und Sie sind bereit, die Rakete zu besteigen. Legen wir los!

Seltsame Ereignisse auf dem Weg zur Kneipe

Mark ist Inhaber des The Splendid Walrus, einer Kneipe mit einem ausgefallenen Namen, die unter den örtlichen Biertrinkern begeisterte Anhängerschaft gefunden hat. Bei Mark gibt es stets ein paar ausgefallene Biere vom Fass, die auf der hauseigenen Website bekannt gemacht werden.

Bevor er seinen Lebenstraum Kneipier in die Tat umsetzen konnte, hat Mark sich seine Brötchen als Webentwickler verdient. Es kostete ihn also keine Mühen, eine ordentliche Website für das Splendid Walrus aufzubauen.



Wenn Handybrowser so toll sind ...

Mark hat die Splendid Walrus-Website vor einigen Jahren gestaltet, als mobiles Surfen noch recht unterentwickelt und ungewöhnlich war. Sie wurde für Desktopbrowser wie Firefox, Internet Explorer und Safari erstellt und mit ihnen getestet.

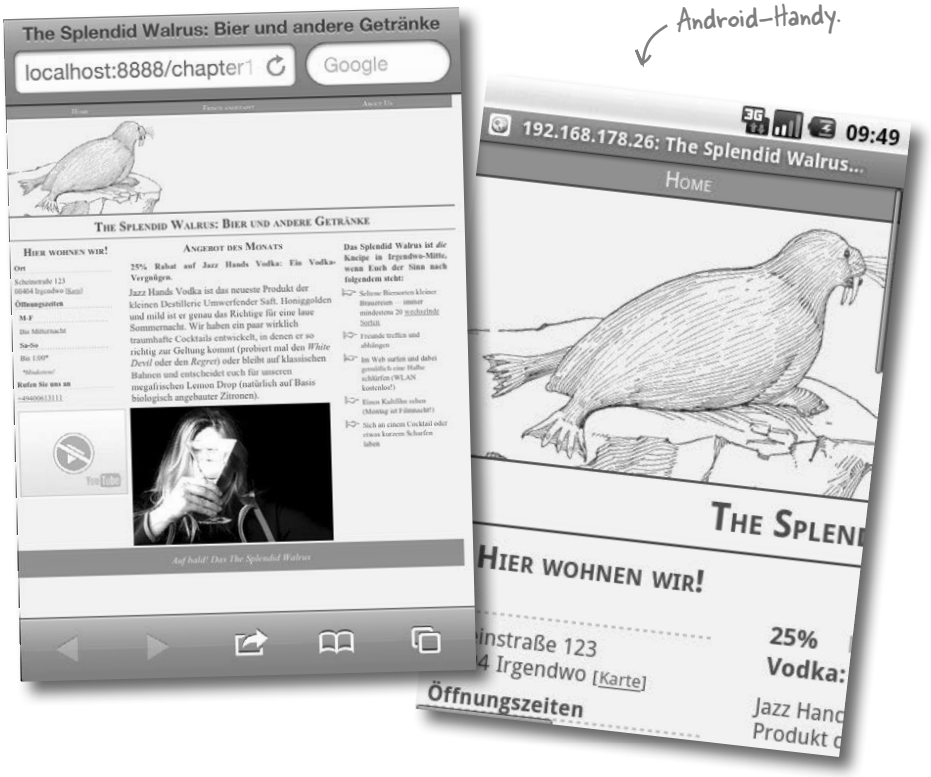
Viele der neueren Mobilbrowser haben einen guten Ruf. Sie werden immer ausgefeilter und mächtiger und beginnen, sich wie einige ihrer Desktopkollegen anzufühlen.

... sollte das doch einfach funktionieren?

Trotzdem erlebte Mark ein böses Erwachen, als er sich die Splendid Walrus-Site auf seinem iPhone 4 ansah. Auch auf dem Android-Gerät eines Freundes sah sie nicht so sonderlich gut aus.

So sieht die Splendid Walrus-Site auf einem iPhone 4 aus ...

... und so auf einem Motorola Backflip-Android-Handy.



Was ist so anders beim mobilen Web?

1 Es gibt einige Milliarden unterschiedlicher mobiler Webbrowser.
Gut, vielleicht nicht ganz so viele, aber wenn man für das mobile Web entwickelt, hat man gelegentlich den Eindruck. Anders als bei Desktopbrowsern, von denen nur ein paar wirklich relevant sind, gibt es Hunderte verschiedener Mobilbrowser. Leider wahr.

Und jedes Mal, wenn Sie meinen, alle im Griff zu haben, taucht irgendwo ein neuer auf.

2 Die Unterstützung für Webtechnologien schwankt erheblich.
Bei älteren Mobilbrowsern (oder neueren auf weniger fähigen Geräten) dürfen Sie sich keine Hoffnung auf eine zuverlässige CSS- oder JavaScript-Unterstützung machen. Einige Dinge werden selbst von den neuesten Browsern nicht unterstützt oder auf irritierend unterschiedliche Weise oder einfach nur fehlerhaft implementiert. Es ist wie im Wilden Westen!

3 Mobilgeräte sind kleiner und langsamer.
Klar, wir wissen, dass die neueste Generation von Mobilgeräten Höchstleistungstaschencomputer sind. Trotzdem verblassen sie, was die Rechenleistung angeht, im Vergleich zu einem Desktoprechner (oder einem Laptop). Mobilnetzwerke können instabil und die Netzabdeckung kann erbärmlich sein, und dann ist die Datenübertragung noch nicht einmal kostenlos oder unbegrenzt. Das heißt, dass wir uns darüber Gedanken machen müssen, wie wir unsere gewaltigen, medienreichen, komplexen Sites auf eine Diät setzen können, die diesen Leistungsgrenzen Rechnung trägt.

4 Mobilschnittstellen fordern andere Benutzerschnittstellen.
Dass Mobilbrowser Desktopwebseiten mit ein paar Macken darstellen können, heißt nicht, dass das auch so sein *sollte*. Der Bildschirm ist kleiner. Die Interaktionen und Erwartungen sind andere.

Menschen mit Mobilgeräten nutzen die unterschiedlichsten Eingabegeräte: Finger, Stifte, die Winztasten auf ihren BlackBerry-Geräten. Eingaben und das Ausfüllen von Formularen sind bestenfalls umständlich. Auf eine Schrift zu schielen, die für ein Desktopbrowserfenster gedacht ist, kann Nutzern Kopfschmerzen bereiten und zu mittleren Wutausbrüchen führen. So langsam sehen Sie vermutlich, worauf wir hinauswollen.

So rendert Marks iPhone 4 die Splendid Walrus-Website. Das sieht nicht gerade cool aus. Erkennen Sie die Problemzonen? Notieren Sie alle Probleme, die Ihnen auffallen.

❶ .....
.....
.....
.....

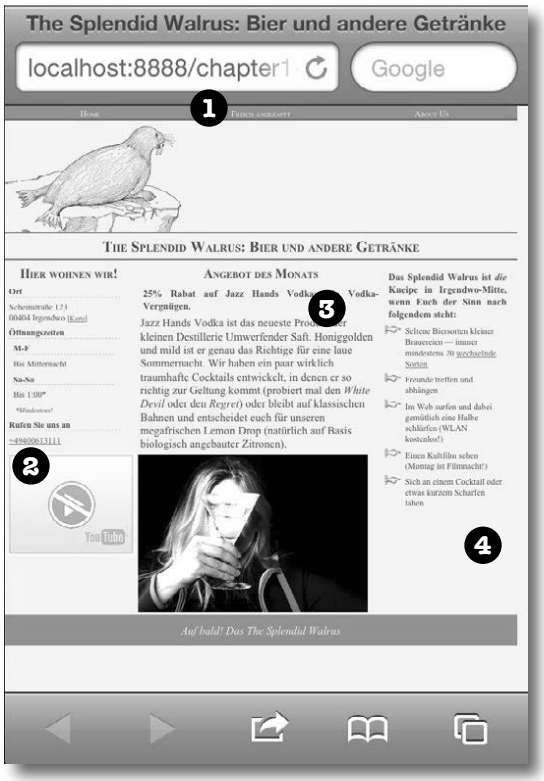
❷ .....
.....
.....
.....

❸ .....
.....
.....
.....

❹ .....
.....
.....
.....

Lösung zur Übung

Haben Sie die Problemzonen erkannt?

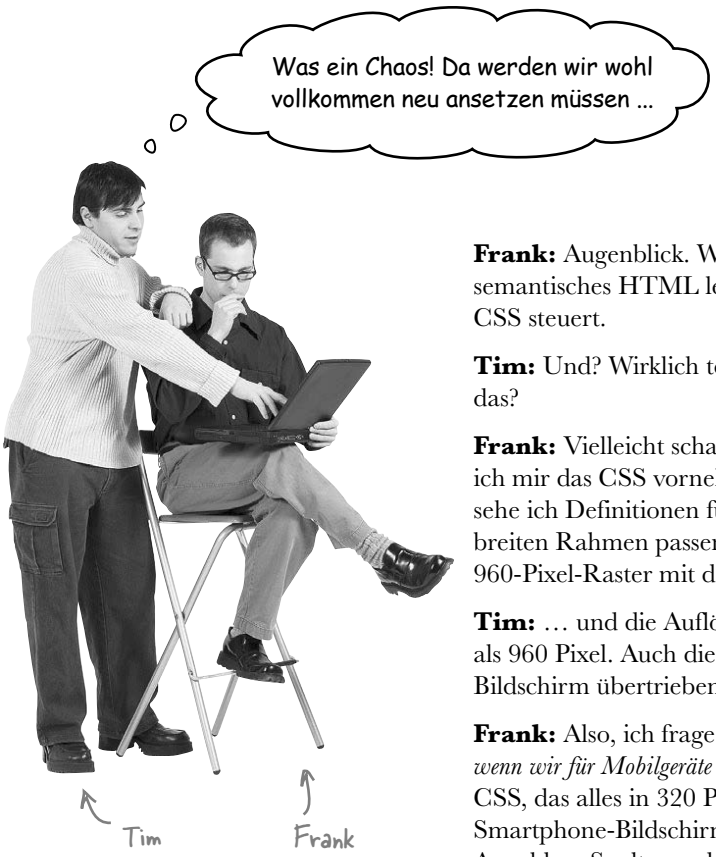


1. Die Navigationslinks sind so winzig, dass man sie weder lesen noch anklicken kann.

2. Das eingebettete YouTube-Video funktioniert nicht.

3. Das Drei-Spalten-Layout sieht bei dieser Bildschirmauflösung gepresst aus, und der Text ist schlecht lesbar.

4. Auf der rechten Seite des Bildschirms ist eine komische Lücke.



Frank: Augenblick. Wir wissen, dass Mark großen Wert auf sauberes, semantisches HTML legt und Layout und Styling so viel wie möglich mit CSS steuert.

Tim: Und? Wirklich toll und auch sehr professionell, aber was bringt uns das?

Frank: Vielleicht schauen wir uns die Sache einfach mal an. Wenn ich mir das CSS vornehme, das er für die Splendid Walrus-Site nutzt, sehe ich Definitionen für Breiten und Größen, die in einen 960 Pixel breiten Rahmen passen. Das scheint doch, als hätte er die Site auf einem 960-Pixel-Raster mit drei Hauptspalten entworfen.

Tim: … und die Auflösung der meisten Mobilgeräte ist erheblich kleiner als 960 Pixel. Auch die drei Spalten scheinen für den viel kleineren Bildschirm übertrieben.

Frank: Also, ich frage mich, ob man nicht vielleicht … *was wäre denn, wenn wir für Mobilgeräte ein anderes CSS nutzen wurden?* Vielleicht ein CSS, das alles in 320 Pixel fügt. Ist das nicht eine Standardbreite für Smartphone-Bildschirme? Und vielleicht sollten wir gleich auch die Anzahl an Spalten reduzieren.

Tim: Gute Idee, Frank. Aber mir ist nicht klar, wie wir das ohne eine Menge Programmierung auf Serverseite erreichen können. Das ist doch die einzige Möglichkeit, Mobilgeräte dazu zu bringen, ein anderes CSS zu nutzen, oder?

Frank: Vielleicht reden wir mal mit Jana. Die ist gerade von der Awesome Cool Mobile Web Camp-Konferenz zurückgekehrt und seitdem völlig aus dem Häuschen wegen etwas, das man *Responsive Webdesign* nennt.

Tim: Wie hätte ich das vergessen können! Sie redet ja von nichts anderem mehr.

Frank: Sie sagt, unter Webentwicklern sei es der letzte Schrei. Und das, was sie sagt, klingt so, als ginge es dabei zumindest in Teilen darum, je nach Situation ein anderes CSS anzuwenden, ohne dass man dazu auf Programmierseite große Schritte unternehmen müsste. Anscheinend ist es vor allem für die Entwicklung mobiler Webseiten geeignet. Ich habe mir nicht alle Einzelheiten gemerkt, aber vielleicht sollten wir uns die Sache einmal ansehen.

Responsive Webdesign

Mit dem Begriff ***Responsive Webdesign (RWD)*** bezeichnet man einen Satz von Techniken, für die sich der Webdesigner Ethan Marcotte stark macht. Sites, die gemäß diesem Verfahren entworfen werden, passen ihre Layouts der Umgebung des Browsers des Benutzers an – hauptsächlich indem sie mit CSS einige elegante Dinge anstellen.

Lesen Sie den Originalartikel von Ethan zu RWD für A List Apart unter http://bit.ly/nRePnj.

In Abhängigkeit von bestimmten Browserbedingungen wie der Fenstergröße, der Gerätausrichtung oder dem Seitenverhältnis können wir unterschiedliches CSS anwenden. Wir müssen überdenken, wie wir das Seitenlayout so gestalten können, dass sich unser altes und als Allzwecklösung eingesetztes Spalten- und -Grid-Layout ohne großen Aufwand den Bedingungen fügt, die durch die vielen unterschiedlichen Fenstergrößen diktiert werden.

RWD ist eines der einfachsten und schnellsten Verfahren, eine Webseite auf einer Vielzahl von Geräten ansehnlich zum Funktionieren zu bringen – und Sie können dabei die Webfertigkeiten einsetzen, die Sie bereits besitzen.

Das Rezept für Responsive Webdesign

Es gibt drei elementare Techniken zur Gestaltung einer responsiv entworfenen Website:

1. **CSS3-Medienabfragen (Media Queries)**
   Bestimmte Aspekte der aktuellen Browserumgebung auswerten, um festzulegen, welches CSS angewandt werden soll.

Wir können in Abhängigkeit von diversen Faktoren wie der Breite des Browserfensters, dem Seitenverhältnis oder der Gerätausrichtung jeweils andere CSS-Regeln anwenden.

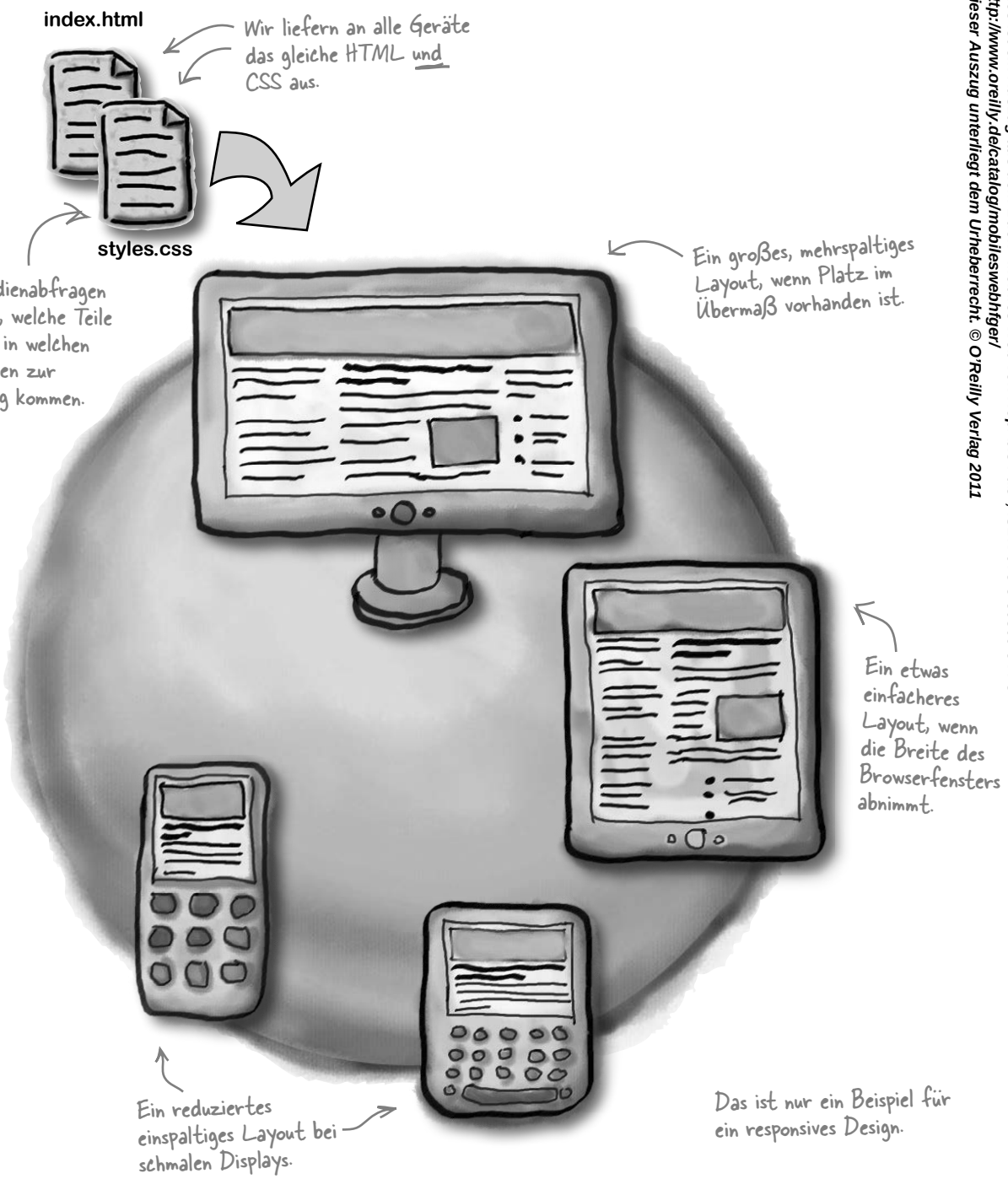
2. **Flüssige Grid-Layouts**
   Relative CSS-Maßangaben statt absoluter Werte für die Elemente des Seitenlayouts verwenden.

RWD nutzt für Spalten und andere Layoutelemente Prozentwerte statt Pixeln.

3. **Flüssige Bilder und Medien**
   Bilder und Medien mit einigen CSS-Tricks dazu bringen, dass ihre Größe an die Beschränkungen ihres Containers angepasst wird.

Flüssige Bilder und Medien berücksichtigen die Grenzen ihrer Elternelemente und ändern ihre Größe im gleichen Verhältnis wie der Rest des Layouts.

Ein Beispiel einer responsiv gestalteten Site



Andere Umgebung, anderes CSS

Wenn Sie schon länger in der Webentwicklung tätig sind (und mit CSS umgehen können), haben Sie mit den CSS-Medientypen vielleicht schon Freundschaft geschlossen. **Wir können @media-Regeln nutzen, um CSS selektiv anzuwenden**.

CSS-Medientypdeklarationen in einer CSS-Datei sehen so aus:

```
@media screen { /* CSS-Regeln für Bildschirme! */ }
```

>>screen<< ist ein Medientyp.

Die Regeln zwischen den geschweiften Klammern werden nur angewandt, wenn der Inhalt auf einem Bildschirm dargestellt wird.

Außerdem können Sie Medientypen in den `<link>`-Elementen Ihrer HTML-Dokumente nutzen, um CSS selektiv anzuwenden.

```
<link rel="stylesheet" type="text/css" href="print.css" media="print" />
```

Die Regeln in diesem Stylesheet werden nur angewandt, wenn der Inhalt auf einem >>Druckgerät<< ausgegeben wird (d.h. einem Drucker).

>>print<< ist ein weiterer Medientyp.

Ein derartiger Verweis auf den print-Medientyp wird häufig genutzt, um ein Druck-Stylesheet zu erstellen – also CSS-Styles, die nur angewandt werden, wenn der Inhalt ausgedruckt wird.

Medientypen unter der Lupe

Wichtige (und nützliche) Medientypen sind unter anderem `screen`, `print` und `all`. Daneben gibt es weniger gebräuchliche Medientypen wie `aural`, `braille` und `tv`.

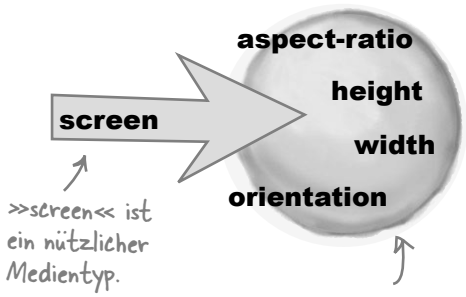
Neugierig? Wenn Sie einer von den Menschen sind, die zum Vergnügen oder zur Befriedigung ihrer Neugier technische Spezifikationen lesen, können Sie sich die Definition aller in CSS2 definierten Medientypen auf der Site des W3C unter *www.w3.org/TR/CSS2/media.html* ansehen.

Medientypen und Medieneigenschaften

Sie haben bestimmte Kennzeichen – wie z. B. Ihr Alter oder Ihre Größe – und Medientypen haben das auch. Und so wie das The Splendid Walrus wohl die Regel aufstellen sollte, dass Gäste, an die Alkohol ausgeschenkt wird, mindestens 16 bzw. 18 Jahre alt sein müssen, sollten wir eventuell bestimmte CSS-Regeln definieren, die nur auf Browserfenster angewandt werden, deren Breite in einen bestimmten Bereich fällt.

Wir haben Glück! Die Breite, `width`, gehört wie `color` und `orientation` zu den in CSS3 für alle gebräuchlichen Medientypen definierten **Medieneigenschaften**. Medien***typen*** haben also Medien***eigenschaften***.

Allein bringen uns Medieneigenschaften noch nicht viel. Wir brauchen eine Möglichkeit, uns vom Browser Informationen zu den Eigenschaften liefern zu lassen, die uns interessieren, und natürlich auch eine Möglichkeit, diesen Informationen entsprechend zu agieren. Das ermöglichen uns ***CSS3-Medienabfragen***.



>>screen<< ist ein nützlicher Medientyp.

Einige der Medieneigenschaften des Medientyps >>screen<<.

P. S. Es gibt noch mehr, aber das sind die, die uns am meisten interessieren.

CSS-Medienabfragen

Mal wieder unser Medientyp >>screen<<!

>>width<< ist die Medieneigenschaft, die wir für den Medientyp >>screen<< auswerten wollen.

Diese CSS-Regeln werden nur angewandt, wenn die Medienabfrage mit WAHR ausgewertet wird.

```
@media screen and (min-width:480px) { /* CSS-Regeln */ }
```

>>min-<< ist ein Medienabfragenpräfix. Es sagt, recht intuitiv, dass wir die Minimalbreite abfragen wollen.

Wenig überraschend gibt es auch ein >>max-<<Präfix.

Das heißt: **Wird das Dokument auf einem Bildschirm wiedergegeben, *UND* ist das Fenster *aktuell* mindestens 480 Pixel breit?**
Ja? Gut, dann wende diese CSS-Regeln an.

Ein weiteres Beispiel:

```
@media print, screen and (monochrome) { }
```

Ein logisches >>ODER<< wird durch ein Komma repräsentiert. Ja, das ist leicht verwirrend.

>>monochrome<< ist eine Medieneigenschaft des Medientyps >>screen<< und ist entweder WAHR oder FALSCH.

Wird dieses Dokument auf einem Drucker ODER einem Bildschirm ausgegeben, der monochrom ist (schwarz und weiß)?
Falls ja, wende diese Styles an!

CSS3-Medienabfragen sind logische Ausdrücke, die die aktuellen Werte der Medieneigenschaften des Browsers des Benutzers auswerten. Wird der Medienabfragenausdruck mit WAHR ausgewertet, wird das eingeschlossene CSS angewandt.

ÜBUNG

CSS-Medienabfragen übersetzen: Testen Sie sich! Ordnen Sie die Medienabfrage der passenden Beschreibung zu.

```
@media all and (orientation: landscape) {}
```

```
<link rel="stylesheet" type="text/css"
href="my.css" media="screen and (color)" />
```

```
@media print and (monochrome) {}
```

```
@media screen and (color) { }
```

Die Regeln in diesem externen Stylesheet bei Farbbildschirmen anwenden.

Diese Styles bei Schwarz-Weiß-Druckern anwenden.

Diese Regeln bei Farbbildschirmen anwenden.

Diese Styles auf alle Medientypen anwenden, wenn das Gerät im Landschaftsmodus ist.

Konnten Sie die Medienabfragen entziffern?

LÖSUNG ZUR ÜBUNG

Ein Medientyp >>all<<, fragen Sie? Ja. Das ist der Typ, den wir nutzen, wenn wir uns die gleiche Medieneigenschaft bei allen Medientypen ansehen wollen.

```
@media all and (orientation: landscape) {}
```

Die Regeln in diesem externen Stylesheet bei Farbbildschirmen anwenden.

```
<link rel="stylesheet" type="text/css"
href="my.css" media="screen and (color)" />
```

Diese Styles bei Schwarz-Weiß-Druckern anwenden.

```
@media print and (monochrome) {}
```

Diese Regeln bei Farbbildschirmen anwenden.

```
@media screen and (color) { }
```

Diese Styles auf alle Medientypen anwenden, wenn das Gerät im Landschaftsmodus ist.

Okay. Das mit den Medienabfragen habe ich jetzt gerafft. Vielleicht kann ich sogar schon eigene schreiben. Aber was mache ich dann? Wie schreibe ich CSS für Mobilgeräte?

CSS: Wie anders ist anders?

Wir haben ein Werkzeug, mit dessen Hilfe wir situationsbedingt unterschiedliches CSS anwenden können. Aber was tun wir jetzt?

Keine Panik. Wir müssen mobilfreundliches CSS schreiben, müssen aber dazu nicht vollkommen bei null anfangen. Und wir werden für unsere Mobilgeräte auch kein vollkommen anderes CSS benötigen – große Teile des bereits vorhandenen CSS können wir wiederverwenden.

Folgendes werden wir tun, um unser mobilfreundliches Layout zu generieren:

- ☐ Wir prüfen das aktuelle Layout von splendidwalrus.com und analysieren seine Struktur.
- ☐ Wir identifizieren die Layoutelemente, die sich ändern müssen, damit sie auf Mobilbrowsern besser funktionieren.
- ☐ Wir generieren mobilfreundliches CSS für die von uns identifizierten Elemente.
- ☐ Wir organisieren unser CSS und wenden das Mobil- und das Desktop-CSS mithilfe von Medienabfragen selektiv an.

Anderes CSS werden wir nur für die Layoutelemente schreiben, die bei Mobilgeräten anders sein müssen.

Die aktuelle Struktur der Splendid Walrus-Site

Werfen Sie einen Blick auf die *index.html*-Datei für die Splendid Walrus-Site im Verzeichnis *chapter1*. Wenn Sie Ihre Vorstellungskraft einsetzen und sich den Inhalt wegdenken, können Sie das klassische Gerüst einer HTML-Seite erkennen:

Den Download-Code finden Sie unter http://examples.oreilly.de/germanexamples/mobileswebhfger/

Die drei Spalten für den Inhalt.

```
<div class="navigation">...</div>
<div class="header">...</div>
<h1>...</h1>
<div id="visit" class="column">...</div>
<div id="points" class="column">...</div>
<div id="main" class="column">...</div>
<div class="footer">...</div>
```

Dass man die rechte Spalte im Markup vor der mittleren Spalte angibt, ist ein alter Trick im Webdesign-Geschäft, der den Umgang mit Layouts vereinfacht, die CSS-Floats nutzen.

Das aktuelle, desktopzentrierte CSS baut die Seite so auf:

Werfen wir einen Blick auf das CSS, das dieses Layout definiert, und finden wir heraus, was sich ändern muss, um die Seite an das Mobilformat anzupassen.

Das aktuelle CSS analysieren

Öffnen Sie die *styles.css*-Datei für die Splendid Walrus-Site.

Am Anfang der Datei steht eine Menge CSS, über das wir uns keine Gedanken machen müssen. Farben, Schriften und das Styling können wir in den Desktop- und Mobilvarianten gemeinsam nutzen.

Was uns interessiert, ist das **strukturelle** CSS gegen Ende der Datei.

Uns interessieren nur die strukturellen Teile der CSS-Datei.

```
/* Struktur */
body, .header, .navigation, .footer {
  width: 960px;
}
.header, .navigation, .footer {
  clear: both;
}
.column {
  margin: 10px 10px 0 0;
}
.navigation {
  min-height: 25px;
}
.navigation ul li {
  width: 320px; /* 960/3 */
}
.header {
  background:url(images/w.png) no-repeat;
  height: 200px;
}
#visit {
  width: 240px;
  float: left;
}
#points {
  width: 240px;
  float: right;
}
#main {
  margin: 10px 260px 0 250px;
  width: 460px;
}
```

Der Body ist 960 Pixel breit. Die Header-, Footer- und Navigationselemente nehmen die volle Breite ein.

Weil diese Elemente die volle Breite einnehmen, ist sichergestellt, dass neben ihnen nichts schwebt.

clear: both sichert, dass diese Elemente auf einer neuen >>Zeile<< beginnen – dass sich also nichts neben ihnen befindet.

Alle Spalten (visit, right und points) haben oben und rechts einen Abstand von 10 Pixeln (d. h. einen >>Bundsteg<<).

Die Navigationslinks stecken in einer <ul>. Sie werden horizontal angeordnet, und jedes <li> nimmt 1/3 der Seitenbreite ein.

Jedes <li> erhält 1/3 der Seitenbreite, weil es drei Links gibt.

Der Header hat ein Hintergrundbild und muss deswegen 200px hoch sein, damit das Bild vollständig zu sehen ist.

Die linke und die rechte Spalte sind jeweils 240 Pixel breit und schweben.

Es scheint, als müsste die Hauptspalte 480 Pixel breit sein (960 minus zwei mal 240, die Pixel der Spalten rechts und links). Aber tatsächlich ist sie nur 460 Pixel breit, da es zwei 10 Pixel breite Bundstege zwischen den Spalten gibt.

Die Hauptspalte nutzt Abstände, um sich selbst zu positionieren, sie schwebt nicht.

Sie wird durch den linken Abstand von 260px und den rechten Abstand von 250px im Fenster positioniert.

Was muss sich ändern?

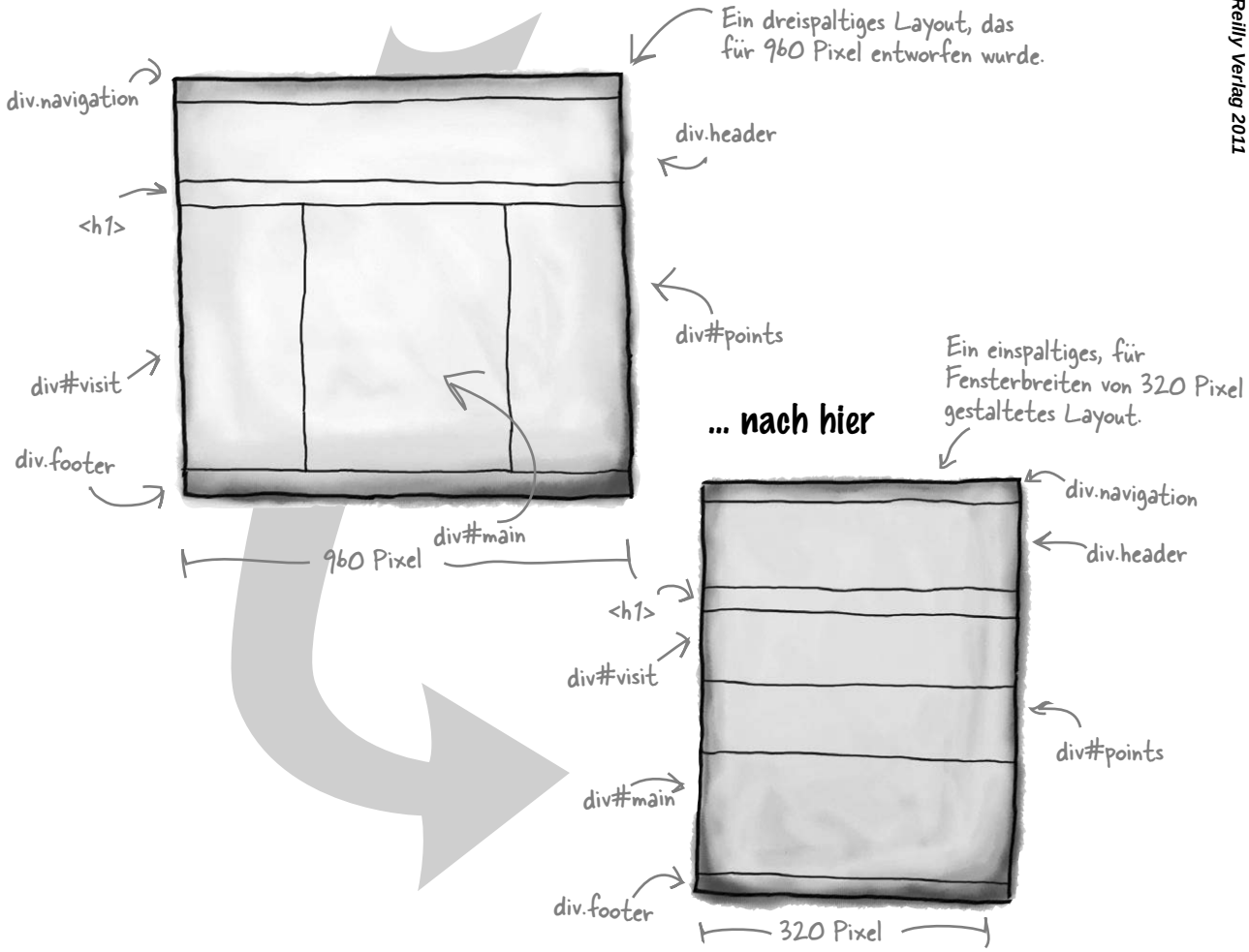
1. **Die Seite und ihre Strukturelemente müssen in 320 Pixel passen.**
   Wie Frank auf Seite 9 erwähnte, sind 320 Pixel eine bei Mobilgeräten gängige Bildschirmbreite.

2. **Die drei Spalten müssen auf eine reduziert werden.**
   Das ursprüngliche für den Desktop konzipierte dreispaltige Layout fühlte sich auf dem Mobilgerät »gequetscht« an.

Wir werden das CSS nicht von Grund auf neu schreiben müssen.

Wir müssen nur einige der strukturellen Layoutelemente anpassen. Der Rest – Typografie, Farben und anderer Kram – kann unverändert bleiben.

Für die mobile Version müssen wir von hier ...



Das CSS identifizieren, das sich ändern muss

Der hervorgehobene Code muss für unsere Mobilversion angepasst werden.

```
/* Struktur */
body, .header, .navigation, .footer {
  width: 960px;
}

.header, .navigation, .footer {
  clear: both;
}

.column {
  margin: 10px 10px 0 0;
}

.navigation {
  min-height: 25px;
}

.navigation ul li {
  width: 320px; /* 960/3 */
}

.header {
  background:url(images/w.png) no-repeat;
  height: 200px;
}

#visit {
  width: 240px;
  float: left;
}

#points {
  width: 240px;
  float: right;
}

#main {
  margin: 10px 260px 0 250px;
  width: 460px;
}
```

styles.css

Wir müssen die Breite der Seite sowie die Header-, Footer- und Navigationselemente anpassen.

Diese Regel brauchen wir in unserer Mobilversion nicht (aber sie stört auch nicht).

Weil in unserem Mobillayout nichts schwebt, brauchen wir keine Clears.

Das ist so in Ordnung: Die Navigationslinks sollen mindestens so hoch sein.

Wir müssen die Breite der Navigationslinks anpassen, damit sie auf den kleineren Bildschirm passen.

Wir müssen die Floats entfernen und die Breite der Visit- und Points-Spalten anpassen.

Die >>Spalten<< werden im Mobillayout vertikal, nicht horizontal angeordnet. Wir fügen zwischen den Spalten (vertikal) einen Abstand ein, entfernen aber den Bundsteg.

Da wir für den Header das gleiche Hintergrundbild nutzen, kann das gleich bleiben.

Es sieht aus, als müssten wir hier die 200px anpassen, aber das tun wir nicht, da wir das gleiche Bild nutzen.

Die Abstände für die Positionierung benötigen wir nicht mehr (#main überspannt die gesamte Breite), und die Breite müssen wir ändern.

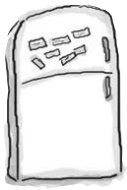
Schritte zur Erstellung des mobilspezifischen CSS

1. **Die Breite in den hervorgehobenen CSS-Regeln anpassen.**

2. **Die CSS-Regeln löschen, die wir nicht mehr benötigen.**

3. **Die gemeinsam genutzten CSS-Regeln auslagern.**

Das werden wir später erledigen.



Mobile CSS-Magneten

Nutzen Sie die Magneten, um das mobilspezifische CSS zu erstellen.

```
body, .header, .footer, .navigation {

}
.column {

  border-bottom: 1px dashed #7b96bc;
}
.navigation ul li {

}
#visit, #points, #main {

}
```

Dieser Rahmen soll einen visuellen Abstand zwischen den (jetzt vertikalen) Spalten einfügen, wenn sie auf der Seite aufgebaut werden.

```
width: 320px;
```

```
width: 106.6667px;
```

```
margin: 10px 0;
```

```
width:320px;
```

Mobile CSS-Magneten, Lösung

```
body, .header, .footer, .navigation {
  width: 320px;
}
.column {
  margin: 10px 0;
  border-bottom: 1px dashed #7b96bc;
}
.navigation ul li {
  width: 106.6667px;
}
#visit, #points, #main {
  width:320px;
}
```

Ta-da! Mobilspezifisches CSS

Das wäre erledigt! Diese vier CSS-Regeln sind alles, was wir an mobilspezifischem Layout benötigen. Jetzt müssen wir dafür sorgen, dass sie von unseren Mobilgeräten auch genutzt werden.

Und wie werden wir das tun? Hier greift uns unsere alte Freundin Frau Medienabfrage unter die Arme! Wir werden gleich eine Medienabfrage konstruieren, um dieses CSS auf Geräte anzuwenden, deren Browserfenster höchstens 480 Pixel breit ist.

Weswegen 480 Pixel? Diese Auflösung hat die >>lange Seite<< (d. h. die Landschaftsausrichtung) vieler beliebter Smartphones.

Verschwunden sind sie nicht ...

… sie müssen in unserem mobilspezifischen CSS bloß nicht mehr vorhanden sein. Warum? Weil die fraglichen CSS-Regeln in beiden Layouts (Desktop und Mobil) die gleichen sind.

Wir werden das gemeinsam genutzte CSS außerhalb der Medienabfragen angeben, damit wir nicht an zwei Stellen die gleichen CSS-Regeln pflegen müssen. Das wollen wir jetzt angehen.

Der Rest unseres strukturellen CSS

Gemeinsames strukturelles CSS

Sehen Sie? Wie gesagt! Nichts vom CSS ist tatsächlich verschwunden. Hier ist das gemeinsame strukturelle CSS, das wir auf Seite 18 identifiziert haben. Es wurde ausgelagert und ist jetzt einsatzbereit.

```
.header, .footer, .navigation {
  clear: both;
}
.header {
  background:url(images/w.png) no-repeat;
  height: 200px;
}
.navigation {
  min-height: 25px;
}
```

Unser strukturelles CSS für den Desktop

Wir brauchen immer noch ordentliches CSS für Desktopbrowser!

Nachdem wir die gemeinsamen strukturellen CSS-Regeln entfernt haben, bleibt dieses für die **desktopspezifische CSS-Struktur**.

Wir werden eine Medienabfrage nutzen, damit dieses CSS nur bei Viewports ab 481 Pixeln angewandt wird.

```
body, .header, .footer, .navigation {
  width: 960px;
}
.column {
  margin: 10px 10px 0 0;
}
.navigation ul li {
  width: 320px; /* 960/3 */
}
#visit {
  width: 240px;
  float: left;
}
#points {
  width: 240px;
  float: right;
}
#main {
  margin: 10px 260px 0 250px;
}
```

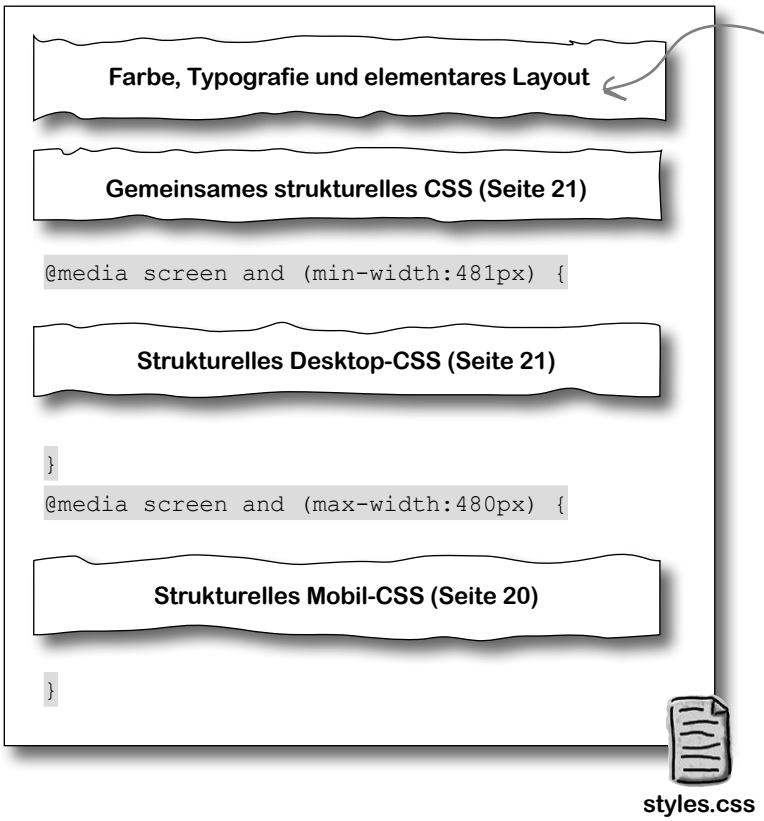
Was kommt jetzt?

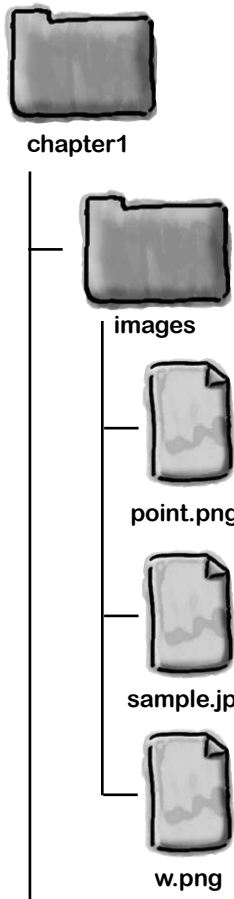
Werfen wir erneut einen Blick auf unsere To-Do-Liste zur Erstellung eines strukturellen CSS, das für Desktop- und Mobilbrowser gleichermaßen funktioniert:

- [x] Wir prüfen das aktuelle Layout von splendidwalrus.com und analysieren seine Struktur.
- [x] Wir identifizieren die Layoutelemente, die sich ändern müssen, damit sie auf Mobilbrowsern besser funktionieren.
- [x] Wir generieren mobilfreundliches CSS für die von uns identifizierten Elemente.
- [ ] **Wir organisieren unser CSS und wenden das Mobil- und das Desktop-CSS mithilfe von Medienabfragen selektiv an.**

Die Teile zusammensetzen

So werden wir die aktualisierte Version von *styles.css* aufbauen.





Eine letzte Sache noch …

Sie müssen in der Datei *index.html* ein **viewport**-<**meta**>-Tag definieren. Mithilfe dieses Tags teilen Sie dem Browser mit, wie »stark gezoomt« der Inhalt dargestellt werden soll. Dieses Kerlchen werden wir uns etwas später ansehen. Für den Augenblick sollten Sie sich einfach auf unser Wort verlassen: Sie werden es benötigen.

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<title>The Splendid Walrus: Bier und andere Getränke</title>
```

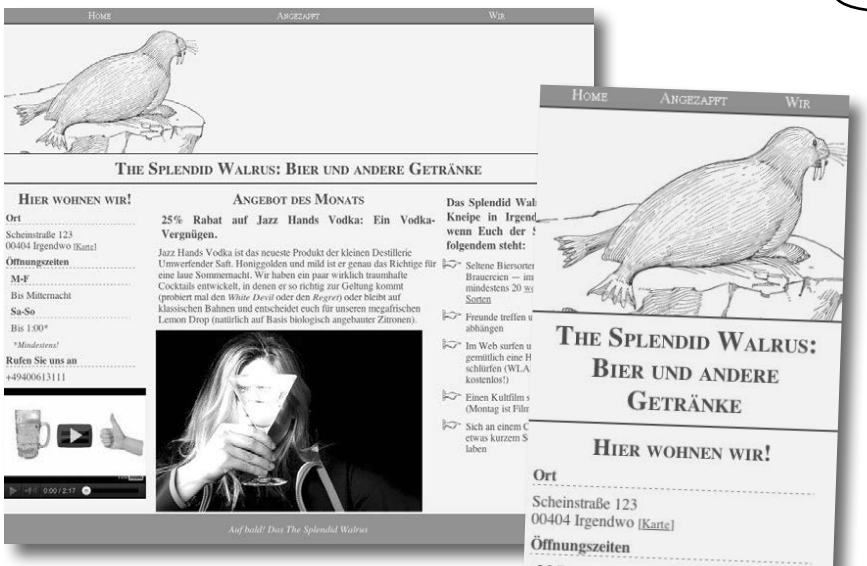
index.html

Tun Sie das!

1. **Bearbeiten Sie die Datei index.html.**
   Fügen Sie das `viewport`-`<meta>`-Tag von Seite 22 darin ein.
2. **Öffnen Sie die Datei styles.css.**
   Sie werden die strukturellen CSS-Regeln gegen Ende der Datei ersetzen. Entfernen Sie die vorhandenen Regeln für strukturelle Elemente.
3. **Fügen Sie die gemeinsamen Regeln ein.**
   Fügen Sie die gemeinsamen strukturellen CSS-Regeln von Seite 21 ein.
4. **Ergänzen Sie das Desktop- und das mobilspezifische CSS.**
   Fügen Sie die Desktopregeln (Seite 21) und die Mobilregeln (Seite 20) ein.
5. **Hüllen Sie das desktop- und das mobilspezifische CSS in Medienabfragen.**
   Fügen Sie die Medienabfragen (Seite 22) ein.

PROBEFAHRT

Laden Sie, nachdem Sie die Änderungen vorgenommen haben, die Datei *index.html* in Ihrem Desktopbrowser und verringern Sie die Breite des Fensters auf weniger als 481 Pixel, um sich das mobilgerechte Layout anzusehen.



In Desktopbrowsern sieht die Seite immer noch gleich aus ...

... bis das Browserfenster schmaler als 481 Pixel ist. Dann können Sie das mobilgerechte Layout sehen!

Das ist zwar ein guter Anfang, aber es gibt noch ein kleines Problem. Wenn ich auf meinem iPhone nach unten scrolle, sehe ich, dass das Bild zu groß für die Seite ist. Es zerstört das Layout.

Frank: Das ist echt frustrierend. Ich dachte, dass sich unser mobilgerechtes CSS auch darum kümmern würde.

Jana: Euer CSS passt das Layout auf den kleineren Bildschirm an, ist aber immer noch recht starr. Schaut euch beispielsweise mal an, was passiert, wenn man das iPhone in Landscape-Ausrichtung dreht.

Frank: Mist. Das Layout ist immer noch 320 Pixel breit … aber der Bildschirm ist 480 Pixel breit. Muss ich etwa für jede denkbare Viewport-Größe anderes CSS schreiben?!

Jana: Ich denke, nein. Genau hierbei sollten uns die Verfahren des **Responsive Webdesigns** helfen. Im Augenblick liefert ihr allen Browsern, deren aktuelles Fenster schmaler als 480 Pixel ist, eine Struktur mit starren Dimensionen, ganz gleich welche Breite das Browserfenster tatsächlich hat. Das ist einfach nicht flexibel genug. Schließlich haben ja nicht alle Mobilgeräte ein 320 Pixel breites Browserfenster.

Responsive Webdesign unterstützt uns dabei, unser Layout anderen Umständen anzupassen. Aber wir müssen die Größe von Elementen ja nicht mit pixelbasierten Maßen vorschreiben, wie wir es jetzt tun. Wir können doch auch ein proportionales Layout nutzen, das sich anderen Nutzungsbedingungen anpasst.

Frank: Proportional? Soll das den Ems und Prozenten und so in CSS entsprechen?

Jana: Ja. So ungefähr. Wir können Prozentwerte statt Pixeln nutzen, wenn wir unser Layout schreiben. Das sorgt dafür, dass sich unser Inhalt dem verfügbaren Platz entsprechend dehnt oder schrumpft wie Wasser, das die Lücken füllt. Deswegen bezeichnet man derartige Layouts häufig als **flüssige Layouts**. Das wird uns, so ganz nebenbei, auch helfen, das eigensinnige Bild auf Kurs zu bringen.

Frank: Dann war die ganze Arbeit mit den Medienabfragen also Zeitverschwendung.

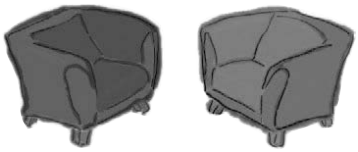
Jana: Auf keinen Fall! Medienabfragen sind ein wichtiger Teil dessen, was Responsive Webdesign möglich macht.

Der nächste Schritt auf dem Weg zu einem responsiven Design – einem, das bequem auf unterschiedlichen Geräten und Browsern zu verwenden ist – besteht darin, dass wir unser starres, pixelbasiertes Layout in ein proportionales, flüssiges Grid-Layout umwandeln.

Dieses Bild ist breiter als 320 Pixel und bricht damit aus dem Layout aus.

Das YouTube-Video funktioniert auf diesem iPhone noch nicht.

Kamingespräche



Heute Abend: **Flüssig vs. Starr**

Flüssiges Grid:

Hi, Starrkopf, du alter Schwede. Ich weiß ja, du treibst dich schon ‘ne Weile rum, aber mal ehrlich, ist deine Philosophie nicht ganz schön problematisch?

Na, bist du nicht so eine Art Relikt, das starr seine Maße definiert und deswegen ziemlich zerbrechlich und unflexibel ist? Responsiv kann man das sicher nicht nennen.

Na, reaktiv und so ... du reagierst einfach nicht auf Änderungen in der Umgebung deiner Nutzer, bleibst immer gleich.

Auf Kosten der Nutzer, Alter. Schau dir doch mal an, was passiert, wenn man dich mit unterschiedlich großen Browserfenstern besucht. Hier wird irgendetwas abgeschnitten, und dort bleibt massenhaft Platz ungenutzt ...

Das mag zu deiner Zeit vielleicht mal funktioniert haben. Heutzutage gibt es schlicht zu viele unterschiedliche Browser und Geräte. Wärst du bereit, den pixelperfekten Layouts abzuschwören – die doch sowieso nur ein Erbe der Tage unserer drucktechnischen Gefangenschaft sind – und deine Inhalte wie Wasser in den vorhandenen Raum fließen zu lassen, dann könntest du dich an die unterschiedlichsten Situationen anpassen.

Starres Grid:

Wie meinen, bitte? Mir war nicht bewusst, dass junge Emporkömmlinge wie Sie etwas von Philosophie verstehen.

Aha! Wieder so ein neudeutsches Wort! Responsiv? Was soll das jetzt wieder heißen?

Ich wüsste nicht, was gegen Tradition einzuwenden wäre. Ich, junger Freund, bewahre, was der Künstler intendierte! 960 Pixel breit, mit 240 Pixel breiten Spalten auf der linken und der rechten Seite.

Wenn das den Gästen nicht gefällt, sollten sie ihren Geschmack korrigieren lassen, zum Friseur gehen und einen Standardbrowser einsetzen.

Na schön, der Herr, dann demonstrieren Sie mir bitte doch, was Sie vermögen.

Was soll an einem Layout mit fester Breite denn falsch sein?

Gäbe es auf der ganzen Welt nur Browser, deren Fenster alle die gleiche Größe hätten, wäre das eine recht sichere, geordnete Welt, in der Designer pixelgenau steuern könnten, wie ihre Webseiten aussehen sollen.

Unglücklicherweise war das Web nie derart einheitlich. Manchmal versuchen wir, uns so zu verhalten, als gäbe es »Standardfensterbreiten« wie 640, 960 oder 1.024 Pixel. Aber eigentlich ist das eine Illusion: Es gibt keine Standardgröße für Browserfenster. Das muss man sich schon eingestehen, bevor man überhaupt beginnt, Mobilgeräte zu berücksichtigen.

Bei einem 960 Pixel breiten Fenster sieht das starre Layout für The Splendid Walrus ansehnlich aus.

Die Site in einem 960-Pixel-Fenster.

Das Layout passt sich anderen Fenstergrößen nicht an

Aber schauen Sie sich an, was bei einem schmaleren Fenster passiert. Die Spaltenbreite bleibt gleich, und das bedeutet, dass Inhalt abgeschnitten wird und der Nutzer deswegen horizontal scrollen muss. Unschön.

Die Site in einem 700-Pixel-Fenster.

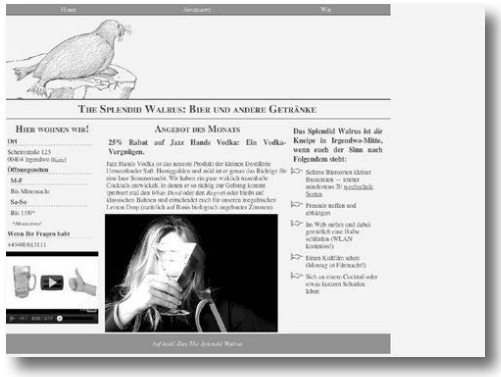
Die rechte Spalte bleibt verborgen und ist nur durch Scrollen erreichbar.

Die linke und die rechte Spalte sind 240 Pixel breit ... immer.

Bei einem breiteren Fenster bleibt das Layout weiterhin 960 Pixel breit und lässt auf der rechten Seite des Bildschirms eine Ödnis mit verschwendetem Platz. Hmmm.

Die Gesamtbreite ist auf 960 Pixel beschränkt.

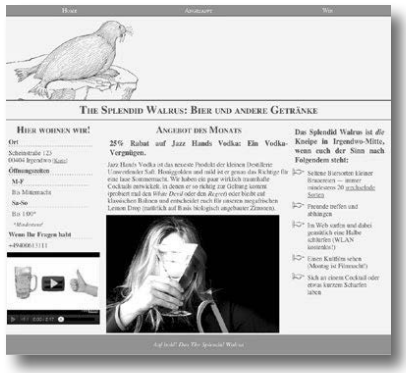
Die Site in einem 1.200-Pixel-Fenster.



Was macht »flüssig« besser?

Flüssige Grid-Layouts nutzen für die Breite proportionale Einheiten (Prozentwerte) anstelle von Pixeln. Wir können der Intention des Designers treu bleiben, dass die linke und rechte Spalte jeweils ein Viertel der Seitenbreite einnehmen, indem wir ihre Breite auf 25 % statt auf 240 Pixel setzen.

Eine flüssige Version des Layouts bei 960 Pixeln. Jetzt sieht die Seite gar nicht groß anders aus, oder?



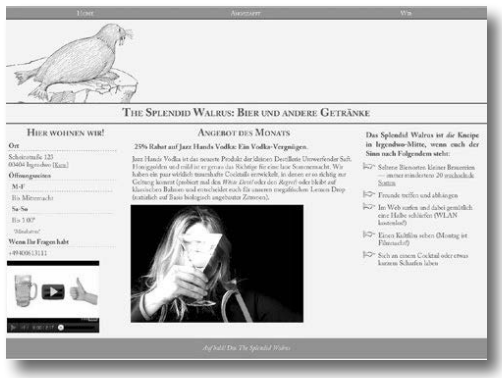
Das Layout passt sich an, wenn sich die Fensterbreite ändert

Bei anderen Fensterbreiten fließt der Inhalt wie Wasser und nimmt den im Layout verfügbaren Raum ein. Die linke und die rechte Spalte nehmen immer 25 % des Fensters ein. Weder wird bei einem schmaleren Fenster der Inhalt beschnitten, noch bleibt bei breiteren Fenstern Raum ungenutzt.

Die flüssige Version des Layouts bei 700 Pixeln.

Die linke und die rechte Spalte sind proportional: 25 % der Breite des Fensters.

Die flüssige Version des Layouts bei 1.200 Pixeln.



KRAFT-TRAINING

Wir werden gleich das Desktop- und das Mobil-CSS von einem starren auf ein flüssiges Grid umstellen. Haben Sie eine Idee, inwiefern uns das helfen könnte, einige der Probleme zu lösen, die Jana bei Mobilbrowsern an der Site entdeckte?

Verflüssigung

Es gibt einiges, was Sie tun müssen, um die Probleme zu bewältigen, die Jana entdeckte, und uns einem responsiven Design anzunähern.

- ☐ Wandeln Sie das pixelbasierte Layout in ein flüssiges Layout um, indem Sie proportionale statt starrer Breiten wählen.
- ☐ Setzen Sie die Standardschriftgröße auf 100 %, damit die Schriften auf der Website ebenfalls proportional skaliert werden.
- ☐ Reparieren Sie das defekte YouTube-Video.
- ☐ Reparieren Sie das Bild, das zu breit ist.

Wenn wir unser Layout flüssig machen wollen, sollten wir auch dafür sorgen, dass die Schriften flexibel sind.

Die Verflüssigungsformel

Nutzen Sie folgende Formel, um ein pixelbasiertes Layout in ein proportionales, flüssiges Layout umzuwandeln:

Die Größe des Elements in Pixeln. → **Ziel**

Kontext = **Ergebnis** ← Der Prozentwert für unsere neue proportionale CSS-Regel.

Die Größe des umschließenden »Kontexts« in Pixeln. → **Kontext**

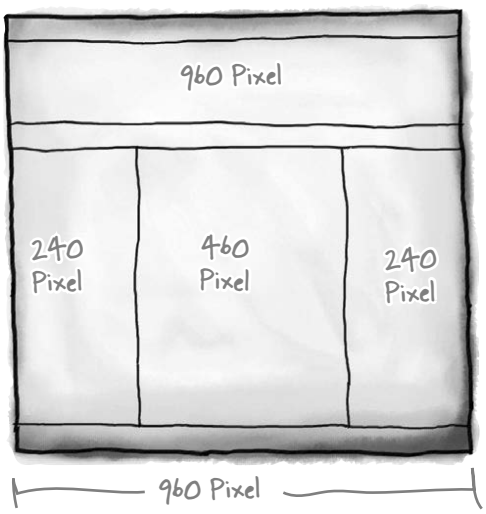
Ein Blick aus der Nähe

Schauen wir uns anhand des Desktoplayouts der Splendid Walrus-Site an, was das bedeutet.

Wir beginnen mit einem **Kontext,** auf dem unsere Proportionen basieren sollen. Das Referenzdesign ist in diesem Fall 960 Pixel breit, und das resultierende flüssige Layout soll die gleichen Proportionen haben wie das aktuelle Design. Wir lassen unsere Berechnungen also auf dieser 960-Pixel-Linie basieren.

Navigation, Header und Footer nehmen alle die vollständige Seitenbreite ein. Ist das der Fall, lässt sich die Verflüssigungsformel tatsächlich ausgesprochen leicht anwenden!

$$\frac{\textbf{960 Pixel}}{\textbf{960 Pixel}} = \textbf{100 \%}$$



Die Verflüssigung fortsetzen

Die linke und die rechte Spalte sollen bei einem übergeordneten Kontext von 960 Pixeln jeweils 240 Pixel breit sein. Das proportionale Verhältnis ermitteln wir erneut mit unserer Verflüssigungsformel:

$$\frac{\textbf{240 Pixel}}{\textbf{960 Pixel}} = \textbf{25 \%}$$

Diese Spalten nehmen ein Viertel der Seitenbreite ein. Das scheint also recht intuitiv, oder?

Bei der Hauptspalte in der Mitte verhält sich das etwas anders. Sie schwebt nicht, sondern wird mithilfe von Abständen positioniert. Aber auch das ist kein Problem. Wir können unsere Verflüssigungsformel ebenfalls einsetzen, um pixelbasierte Abstandsmaße in Prozentwerte umzuwandeln:

$$\frac{\textbf{250 Pixel}}{\textbf{960 Pixel}} = \textbf{26.0416667 \%}$$

Unsere aktuelle CSS-Regel.

```
#main {
  margin: 10px 260px 0 250px;
}
```

$$\frac{\textbf{260 Pixel}}{\textbf{960 Pixel}} = \textbf{27.0833333 \%}$$

Unsere aktualisierte CSS-Regel

```
#main {
  margin: 10px 27.0833333% 0 26.0416667%;
}
```

Schließen Sie die Umwandlung der strukturellen Desktop-CSS-Regeln in proportionale Breiten in der Datei *styles.css* ab. Sie werden alle sechs CSS-Regeln in der Medienabfrage für Fenster, die breiter als 481 Pixel sind, anpassen müssen.

Es gibt keine Dummen Fragen

F: Warum haben diese Zahlen so viele Nachkommastellen? Brauchen wir die wirklich alle?

A: Wir demonstrieren hier das puristische Verfahren, ein flüssiges Grid-Layout zu konstruieren, bei dem die CSS-Maße genau so eingegeben werden, wie sie der Taschenrechner ausspuckt.

In Wirklichkeit runden Browser solche Zahlen, jedoch – beunruhigenderweise – alle auf leicht andere Weise.

Eigentlich liegt das also in Ihrem Ermessen. Sie können die Zahlen auch auf eine oder zwei Nachkommastellen abrunden. Sie können in Ihrem Layout aber auch etwas Spiel lassen – ein oder zwei Prozent, die nicht zugeteilt werden. Ihr Raster ist dann nicht so genau, aber Sie vermeiden einige der Probleme, die die Rundung aufwerfen kann, und machen Ihre Berechnungen weniger streng.

F: Moment. Warum ist der obere Abstand auf `.main` immer noch 10 px? Ist das nicht falsch?

A: Das vertikale Layout ist eine ganz andere Sache als das horizontale. Dabei können wir den 960-px-Kontext nicht nutzen, weil die Höhe des Designs nie eine bekannte Größe ist. Vertikale Layouts mit Prozentwerten sind eine verzwickte Angelegenheit und werden von unterschiedlichen Browsern unterschiedlich (und manchmal nur jämmerlich) unterstützt. Bei vertikalen Abständen ist es in Ordnung, wie hier Pixel zu verwenden.

LÖSUNG ZUR ÜBUNG

Werfen Sie einen Blick auf dieses flüssige Desktop-CSS.

```
@media screen and (min-width: 481px) {
  body, .header, .footer, .navigation {
    width: 100%;
  }
  .column {
    margin: 10px 1.04166667% 0 0;
  }
  .navigation ul li {
    width: 33.333333%;
  }
  #visit {
    width: 25%;
    float: left;
  }
  #points {
    width: 25%;
    float: right;
  }
  #main {
    margin: 10px 27.0833333% 0 26.0416667%;
  }
}
```

960 / 960 = 100 %

10 / 960 = 1.04166667 %

320 / 960 = 33.333333... %

240 / 960 = 25 %

Das haben wir auf Seite 29 erläutert.

Die proportionale Breite rechnen wir also einfach aus, indem wir die Breite des Elements durch die Gesamtbreite des Layouts teilen?

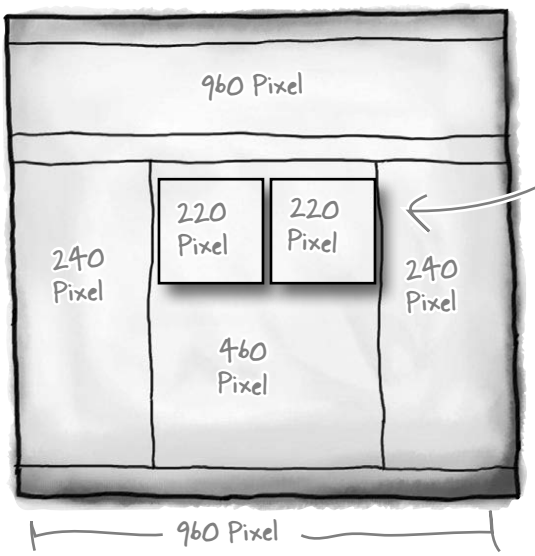
Nicht so schnell!

Manchmal kann sich der »Kontext« ändern …

Kontextwechsel

Ich werde in Zukunft einige neue Bier-Specials anbieten und möchte die auf der Site vorstellen – könnt ihr das Design entsprechend anpassen?

Mark bietet ein neues Monats-Special an, das auf der Splendid Walrus-Site angekündigt werden soll. Statt des Texts mit dem Bild sollen in der Hauptspalte jetzt nebeneinander schwebend die Etiketten von zwei besonderen Biersorten angezeigt werden. In unserem pixelbasierten Referenzdesign sieht das so aus:



Man ist schnell versucht, die Bildbreiten mit folgender Formel in flüssige Proportionen umzurechnen:

$$\frac{\textbf{220 Pixel}}{\textbf{960 Pixel}} \Rightarrow \textbf{22.916667 \%}$$

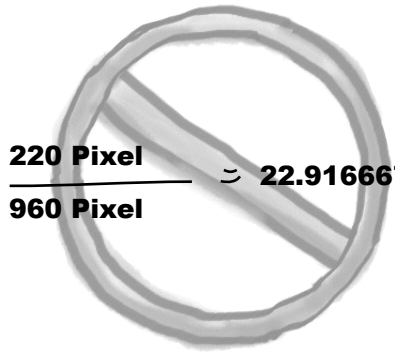
Erscheint Ihnen diese Formel korrekt?

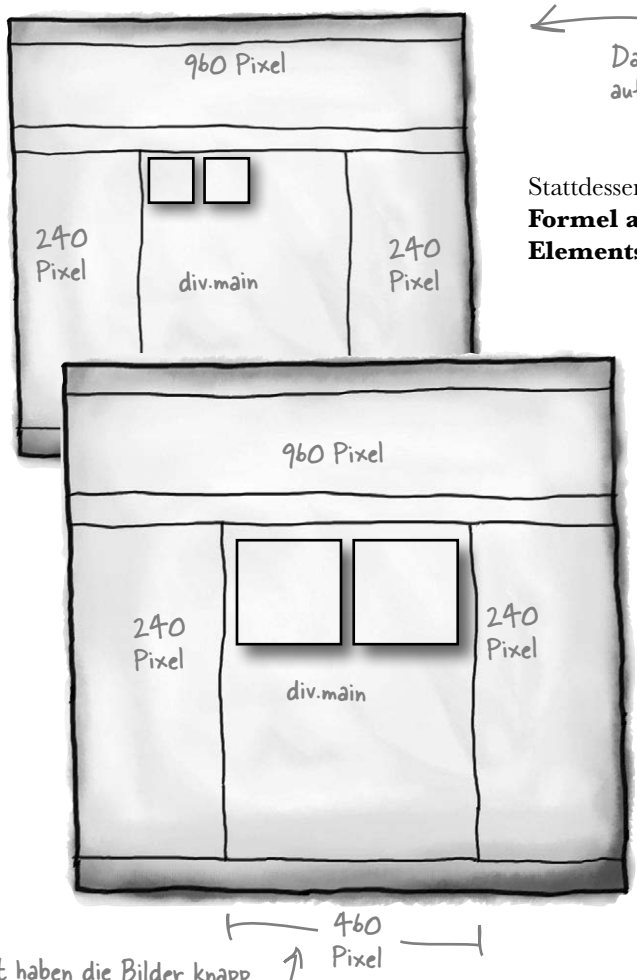
Suchen Sie den Fehler im Bild?

Der Kontext hat sich geändert!

Setzen wir die Bilder so, dass sie 22.9166667 % einnehmen, nehmen sie tatsächlich 22.916667 % ein – 22.916667 % *des umschließenden Elements*.

Das umschließende Element dieser Bilder ist nicht `body` (100 % Breite oder 960 Pixel in unserem Referenzdesign), es ist das `div#main` mit einer Breite von ungefähr 460 Pixeln (proportional ausgedrückt: 47.91667 % von 960 Pixeln). Wir hätten den Bildern damit also gesagt, dass sie etwas weniger als 23 % von 460 Pixeln einnehmen sollen, und der Wert ist erkennbar zu klein!





Das erhalten Sie, wenn Sie width für img.label auf 22.916667% setzen. Nicht schön.

Jetzt haben die Bilder knapp weniger als 50 % der Breite von div#main – das sieht besser aus.

Stattdessen **müssen wir den Kontext in unserer Formel auf die Breite des umschließenden Elements setzen,** und die ist hier 460 Pixel.

Tun Sie das!

220 Pixel / 460 Pixel = 47.826087 %

Neuer Kontext: die Breite von div#main, dem umschließenden Element.

Bildbreiten in Prozentwerten setzen? Es stellt sich heraus, dass das der richtige Weg zur Behebung eines unserer Probleme mit dem Mobillayout ist. Erinnern Sie sich, dass das Foto viel zu groß war und die Seitenbreite ruinierte? Wir können eine Abwandlung dessen, was wir hier nutzen, einsetzen, um das zu reparieren!

Flüssige Bilder

und Medien!

In diesem kleinen CSS-Juwel steckt eine große Macht:

```
img, object {
  max-width: 100%;
}
```

Kawumm!

Mit dieser kleinen Ergänzung verhindern wir, dass ein Bild oder ein eingebettetes Medienobjekt breiter wird als das umschließende Element. Weil sie auf 100 % Breite begrenzt sind – 100% der Breite des umschließenden Elements –, gehorchen Bilder und Medien ihren Eltern und versuchen nicht mehr, aus den Grenzen auszubrechen. Sehr nett!

Flüssige Bilder und Medien skalieren wie ein flüssiges Grid proportional mit dem Layout.

Ein trauriger Abschied ...

Leider muss man für die meisten angenehmen Dinge irgendwo auch immer ein kleines Opfer bringen. Wenn wir diese Verflüssigungstechnik für Bilder und Medien nutzen wollen, müssen wir zwei alten Freunden den Rücken kehren: den Attributen `width` und `height`.

Die CSS-Regel oben überschreibt ein eventuelles `width`-Attribut, wirkt sich aber nicht auf `height`-Attribute aus. Wenn wir die Attribute `height` und `width` nutzen, kann es also passieren, dass wir am Ende ein Bild haben, dessen Breite skaliert wird, während die Höhe unverändert bleibt. Endresultat: ein jämmerlich gequetschtes Bild mit dem falschen Seitenverhältnis.

Auch wenn es einige Möglichkeiten gibt, das zu umgehen, und man nicht gerade mit Freuden auf den Einsatz dieser beiden Attribute verzichtet, werden wir `height` und `width` zunächst einmal meiden.

Flüssige Bilder sind kein Freifahrtschein.

Dass ein Bild auf einem schmaleren Bildschirm sauber skaliert wird, heißt nicht, dass es nicht im Herzen immer noch ein großes Bild ist. Ein 800-KB-JPEG bleibt ein 800-KB-JPEG, selbst wenn es in eine 120 Pixel breite Spalte gequetscht wird.

In Kapitel 2 werden wir uns Techniken ansehen, mit deren Hilfe man unterschiedliche Bilder an unterschiedliche Geräte und Browser ausliefern kann, um so ansonsten verschwendete Bandbreite und Rechenkraft (die für die eigentliche Skalierung erforderlich ist) zu sparen.

Trotzdem bleibt das eine mächtige Technik, die man auf alle Fälle in sein Repertoire aufnehmen sollte.

Haben wir es jetzt geschafft?

Wir nähern uns einem responsiven Design, das sich an mehr Geräte anpasst. Aber es bleiben immer noch einige Dinge zu tun:

Wir müssen noch das Mobil-CSS umwandeln (das Desktop-CSS haben wir bereits erledigt).

- ☐ Wandeln Sie das pixelbasierte Layout in ein flüssiges Layout um, indem Sie proportionale statt starrer Breiten wählen.
- ☐ Setzen Sie die Standardschriftgröße auf 100 %, damit die Schriften auf der Website ebenfalls proportional skaliert werden.
- ☐ Reparieren Sie das defekte YouTube-Video.
- ☑ Reparieren Sie das Bild, das zu breit ist.

Das haben wir mit der Bildverflüssigungstechnik erreicht.

Das Mobil-CSS aufmöbeln

Wir müssen nur ein paar der mobilspezifischen CSS-Regeln umwandeln, um unser Layout flüssig zu machen.

aktuell (starr)

```
@media screen and (max-width:
  body, .header, .footer, .navi
  {
    width: 320px;
  }
.column {
   margin: 1em 0;
   border-bottom: 1px dashed #
}
.navigation ul li {
  width: 106.6667px;
}
#visit, #points, #main {
  width: 320px;
}
```

aktualisiert (flüssig)

```
@media screen and (max-width: 480px) {
  body, .header, .footer, .navigation {
    width: 100%;
  }
.column {
   margin: 1em 0;
   border-bottom: 1px dashed #7b96bc;
}
.navigation ul li {
  width: 33.333333%;
}
#visit, #points, #main {
  width: 100%;
}
```

Moment! Nachdem wir diese beiden Regeln proportional gemacht haben, entsprechen sie genau den Regeln im Desktop-CSS (siehe Seite 30). Wenn wir sie zu den gemeinsamen strukturellen Regeln packen, müssen wir sie nicht doppelt vorhalten.

Details, Details

Kümmern wir uns um die paar Details, die noch verbleiben, bevor unsere aktualisierte Version der Splendid Walrus-Site absolut responsiv ist.

Flexible Schriften einrichten

Bislang ist nur das Layout adaptiv, während die Schriften schwerfällig und steif bleiben. Wie Prozentwerte das flüssige Ying für das Yang starrer Pixelwerte sind, so sind **Ems** die proportionale Einheit für Schriftgrößen. Mark hat in seinem ursprünglichen CSS Ems verwendet, deswegen müssen wir dem <body>-Element nur noch die folgende Regel hinzufügen, um ganz genau zu sein:

```
body {
  background: #f9f3e9;
  color: #594846;
  font: 100% "Adobe Caslon Pro",
  "Georgia", "Times New Roman", serif;
}
```

Diese Rücksetzung der Basisschriftgröße ist ein CSS-Äquivalent zum Urmeter oder zu einer Atomuhr: Damit wird eine explizite Referenz definiert, auf deren Basis andere Schriftgrößen im CSS bestimmt werden. Keine große Sache also, es sorgt nur dafür, dass alles im Maße bleibt!

Schriftgrößen unter der Lupe

Mit dieser Bearbeitung der CSS-Regel für das <body>-Element setzen wir die Basisschriftgröße für die Seite auf 100 %. Aber was heißt 100 %? Hier ist eine Ad-hoc-Daumenregel (und eine *Nährung*):

1em = 100 % ≈ 12pt ≈ 16px

Aber erinnern wir uns daran, dass wir danach streben, unseren Inhalt an die Umgebung des Benutzers anzupassen. Wenn ein Benutzer die Schriftgröße seines Browsers geändert hat, repräsentieren 100 % eine andere absolute Größe.

Denken Sie auch daran, dass Schriften auf Mobilgeräten eine komplexe Geschichte sind und dass 1em in einigen Fällen einer (erheblich) anderen Punkt- oder Pixelgröße entspricht.

Das YouTube-Video reparieren

Viele Mobilgeräte unterstützen Adobe Flash nicht. Das Markup für das eingebettete YouTube-Video ist veraltet: YouTube bietet mittlerweile ein iframe-basiertes Einbettungs-Snippet, das ebenso gut auf einem iPhone (und anderen modernen Geräten) funktioniert. Wir müssen die Datei *index.html* bearbeiten und den aktuellen Einbettungscode ersetzen.

Diese Technik ist nicht auf Flash beschränkt! Andere Medien können ebenfalls auf diese Weise flüssig gemacht werden.

Statt dieser Nur-Flash-Version ...

```
<object width="230" height="179"
type="application/x-shockwave-flash"
data="http://www.youtube.com/v/O-
jOEAufDQ4?fs=1&amp;hl=en_US&amp;rel=0"><embed
src=... /></object>
```

... nutzen wir das!

```
<iframe src="http://www.youtube.com/embed/O-
jOEAufDQ4" style="max-width:100%"></iframe>
```

YouTubes neuerer Einbettungscode ermittelt das zu nutzende Videoformat in Abhängigkeit vom Browser. Er kann für Geräte – wie Marks iPhone – HTML5-Video statt Flash anbieten. Wir haben einfach dieses neuere Snippet aus dem »Einbetten«-Abschnitt auf der YouTube-Seite für dieses Video eingesetzt.

Die Verantwortung nicht vergessen

Wie die Anpassung der Größe flüssiger Bilder die Größe der Bilddatei nicht ändert, so ändert die Verflüssigung von Medien die tatsächliche Größe des tatsächlichen Mediums nicht. Die Entscheidung, ob der Einsatz eines Videos (oder anderer Multimedia-Inhalte) auf einem Mobilgerät die Bandbreite und die Rechenkraft wert ist, die für die Wiedergabe erforderlich sind, unterliegt Ihrer Verantwortung.

Lange Übung

Gut! Bringen wir nun alle diese Änderungen unter, um der Splendid Walrus-Site eine mobilfreundliche Site beiseite zu stellen, die Techniken des Responsive Webdesign nutzt.

Bearbeiten Sie die Datei styles.css!

1. Bearbeiten Sie die CSS-Regeln in der Medienabfrage für Geräte mit geringer Auflösung. Machen Sie diese Breiten proportional (Seite 34).

2. Identifizieren Sie die strukturellen CSS-Regeln, die die Mobil- und die Desktopvarianten gemeinsam haben, nachdem wir sie proportional gemacht haben (Seite 34). Entfernen Sie diese Regeln aus den medienabfragespezifischen Abschnitten und setzen Sie sie in den gemeinsamen strukturellen Abschnitt der CSS-Datei ein.

3. Fügen Sie das CSS von Seite 33 ein, um flüssige Bilder (und Medien) zu implementieren.

4. Aktualisieren Sie die CSS-Regeln für das `<body>`-Element, um eine proportionale Basisschriftgröße zu definieren (Seite 35).

Bearbeiten Sie die Datei index.html!

5. Ersetzen Sie die reine Flash-Einbettung des YouTube-Videos durch die raffiniertere iframe-Variante auf Seite 35.

Schauen Sie sich das Ergebnis an!

Versuchen Sie, die Größe des Browserfensters zu ändern, und beobachten Sie, wie sich der Inhalt anpasst.

6. Speichern Sie die Änderungen, die Sie vorgenommen haben, und laden Sie die *index.html*-Seite in einem beliebigen Webbrowser.

Es gibt keine Dummen Fragen

F: Gut. Mir ist bekannt, dass man Ems in CSS als Einheit nutzt. Aber verstehen tue ich das nicht. Warum macht man das?

A: 1em ist – ganz simpel – eine Repräsentation der aktuellen Schriftgröße im aktuellen Kontext. Das klingt nicht sonderlich aufregend. Spannend wird das erst, wenn Sie Ihre Schriftgrößen in Bezug darauf definieren. Setzen Sie Ihre `<h1>`-Elemente so, dass sie mit 1.5em angezeigt werden, hat die Schrift 150 % der Basisschriftgröße ihres umschließenden Elements.

Sie können sogar die gleiche Verflüssigungsformel nutzen, um starre Schriftgrößen in flüssige, Em-basierte Schriftgrößen zu überführen. Eine flüssige Version einer 18 Punkt großen Schrift in einem Kontext, dessen Basisschriftgröße 16 Punkt ist, berechnen Sie über `18 / 16 = 1.125em` (Ziel / Kontext = Ergebnis).

F: Warum sind das 112,5em und nicht 112,5 %?

A: Das ist im Wesentlichen eine Konvention und dient der Klarheit, aber erfahrungsgemäß ist es auch so, dass Ems plattformübergreifend etwas besser funktionieren. Beim Webdesign ist es üblich, dass man die Breite von Blockelementen mit Prozentwerten angibt, Schriftgrößen hingegen mit Ems. Sieht man von einigen wirklich winzigen Ausnahmen ab, sind Prozentwerte und Ems bei Schriftgrößen äquivalent. Was diese winzigen Ausnahmen betrifft: Die räumen wir aus dem Weg, indem wir `font-size` für das `<body>`-Element auf 100 % setzen.

F: Gibt es neben dem mobilen Web noch andere Gründe dafür, CSS-Medienabfragen zu nutzen?

A: Absolut. Ein Beispiel: Mobilgeräte stellen häufig das untere Ende des Spektrums der Bildschirmauflösungen dar, so mancher moderner Widescreen-Monitor oder Fernseher mit einer sehr hohen Auflösung das obere Ende. Unter Umständen kann es sinnvoll sein, ein Layout für derartige Fensterbreiten anzupassen – beispielsweise indem man weitere Spalten ergänzt.

F: Was hat eigentlich zu der seltsamen Lücke auf der rechten Seite des Bildschirms in der Übung auf Seite 7 geführt?

A: Das ist ein iPhone-spezifisches Problem. Wird eine Webseite »herausgezoomt« dargestellt (wenn also das Verhalten eines Desktopbrowsers genutzt wird), geht Mobile-Safari auf dem iPhone davon aus, dass der Viewport eine Breite von 980 Pixeln hat. Vor der Optimierung war das Splendid Walrus-Layout 960 Pixel breit, und das führte zu dieser unansehnlichen, 20 Pixel breiten Lücke auf der rechten Seite.

F: Was bedeutet die »3« in den CSS3-Medienabfragen?

A: Es gibt mehrere CSS-Versionen. Man würde gern sagen, dass es drei sind, ganz so einfach ist die Lage jedoch nicht. CSS2, das bereits 1998 als »Recommendation« oder »Empfehlung« veröffentlicht wurde, ist die Variante, mit der die meisten Webentwickler seit Langem vertraut sind. Medientypen wurden in CSS2 eingeführt.

CSS3 unterscheidet sich von den früheren CSS-Versionen darin, dass es modularisiert ist. Es gibt etwa 40 verschiedene Module statt einer riesigen, komplexen Spezifikation. Glücklicherweise ist das Media Queries-Modul eines der vollständigeren und stabileren Module.

F: Wenn CSS3 noch nicht fertig ist, wird es denn dann von Browsern unterstützt?

A: Wie wir sagten: Es ist ein Dschungel. Die Unterstützung für die vollständigeren Teile von CSS3 nimmt zu, wie wir später sehen werden, ist es allerdings nichts, was man im Mobilsektor einfach so voraussetzen könnte.

F: Warum steht die rechte Spalte im Mobillayout vor der Hauptspalte?

A: Der `div#points`-Inhalt kommt im HTML-Markup vor dem `div#main`-Inhalt. Im Desktoplayout werden Floats genutzt, um den `#points`-Inhalt so zu positionieren, dass er rechts des `#main`-Inhalts erscheint. Das Mobillayout nutzt keine Floats, und der Inhalt wird deswegen in der Abfolge angezeigt, in der er im HTML erscheint.

F: Sie haben die @import-Syntax für Medientypen übersprungen. Kann ich die @import-Syntax bei Medienabfragen nutzen?

A: Die @import-Syntax ist nicht sonderlich beliebt – sogar so unbeliebt, dass wir sie nicht einmal erwähnt haben. Aber natürlich kann man mit ihr auch Stylesheets auf Basis von Medientypen und Medienabfragen einschließen.

Lange Übung Lösung

Werfen wir einen Blick auf die resultierende CSS-Datei und schauen wir uns unsere Änderungen an.

Das steht unmittelbar am Anfang der CSS-Datei.

```
body {
  background: #f9f3e9;
  color: #594846;
  font: 100% "Adobe Caslon Pro", Georgia,
    "Times New Roman", serif;
}
```

Diese Regeln zeigen wir hier nicht, um Platz zu sparen.

... gemeinsame Typografie, Farben, Rahmen usw. (gemeinsames nicht strukturelles CSS) ...

Gemeinsames strukturelles CSS.

```
.header, .footer {
  clear: both;
}
.header {
  background:url(images/w.png)
no-repeat;
  height: 200px;
}
.navigation {
  min-height: 25px;
}
img, object {
  max-width: 100%;
}
.navigation ul li {
  width: 33.333%;
}
.header, .footer, .navigation {
  width: 100%;
}
```

Und jetzt das auflösungsspezifische strukturelle CSS.

Strukturelles CSS für größere Browserfenster (d. h. Desktop).

```
@media screen and (min-width:481px) {
  .column {
    margin: 10px 1.04166667% 0 0;
  }
  #visit {
    width: 25%;
    float: left;
  }
  #points {
    width: 25%;
    float: right;
  }
  #main {
    margin: 10px 27.0833333% 0 26.0416667%;
  }
}
```

Strukturelles CSS für kleinere Browserfenster (d. h. Mobilgeräte).

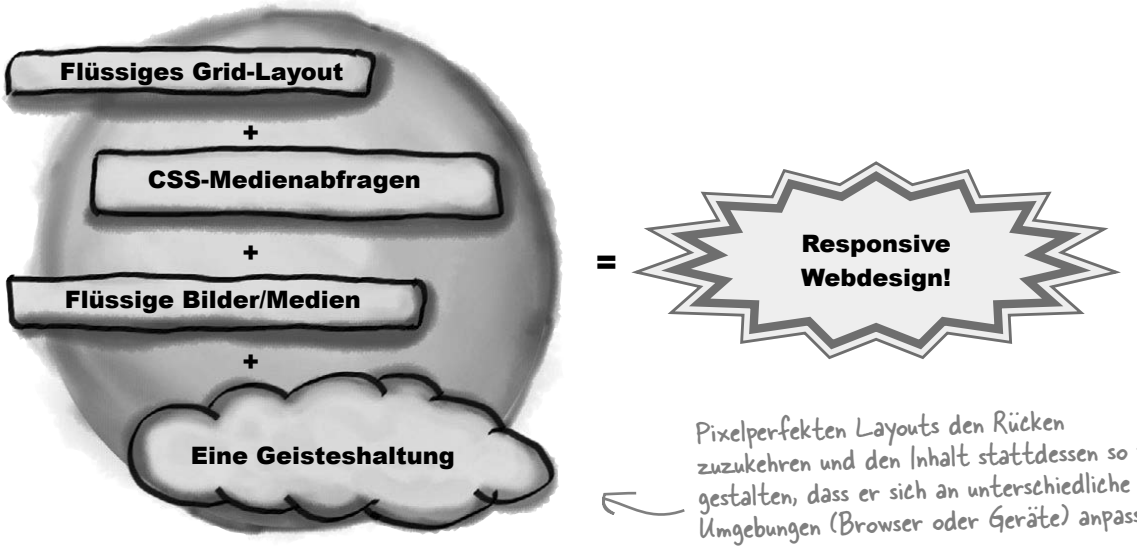
```
@media screen and (max-width:480px) {
  .column {
    margin: 1em 0;
    border-bottom: 1px dashed
#7b96bc;
  }
  #visit, #points, #main {
    width: 100%;
  }
}
```

Das ist eine responsive Site!

KOPF-NUSS

Wir nutzen in unserem mobilfreundlichen CSS das gleiche Hintergrundbild für den Header, obwohl dieses erheblich beschnitten wird. Haben Sie eine Idee, wie man RWD-Techniken nutzen könnte, um ein *anderes* Bild zu verwenden (um Bandbreite zu sparen und die Leistung zu verbessern)?

Responsives Design ist auch eine Geisteshaltung



Punkt für Punkt

- Das mobile Web hat gewisse Ähnlichkeiten mit dem Wilden Westen – es steckt voller Überraschungen und Abenteuer. Die Browserlandschaft des mobilen Webs ist vielfältig und manchmal zum Verrücktwerden.
- Dass man auf einem Mobilgerät das gleiche Layout nutzen kann wie in einem »traditionellen« Browser, heißt nicht, dass man das auch tun sollte.
- **Responsive Webdesign** (RWD) ist eine Sammlung von Verfahren, den Webinhalt an die Bedürfnisse des Nutzers anzupassen – nicht andersherum (den Benutzer zu zwingen, starr formatierte Seiten zu ertragen).
- RWD ist eine Kombination aus **CSS3-Medienabfragen**, **flüssigen Grid-Layouts** und **flüssigen Bildern**. Außerdem ist es eine Weise, über Layout und Inhalt nachzudenken.
- Mit **CSS3-Medienabfragen** kann man CSS unter Berücksichtigung der aktuellen Werte bestimmter Medieneigenschaften selektiv an unterschiedliche Nutzungsumgebungen anpassen.
- **Medientypen** (z.B. `screen`, `print`, `projection`) haben **Medieneigenschaften** (`width`, `color`, `monochrome`, `orientation`). Es sind diese Medieneigenschaften, die wir in unseren Medienabfragen auswerten.
- Eine **CSS-Medienabfrage** ist ein logischer Ausdruck. Wird der Ausdruck zu `TRUE` ausgewertet, werden die eingeschlossenen CSS-Regeln angewandt.
- Ein **flüssiges Layout** ist ein Layout, das statt starrer Breiten proportionale Breiten nutzt, damit sich der Inhalt der Seite anpasst und auf einer Vielzahl von Fensterbreiten natürlich fließt.
- **Flüssige Bilder** sind eine CSS-Technik, die verhindert, dass zu große Bilder (oder Medien) aus ihren Elternelementen ausbrechen, wenn das Elternelement schmaler als das Bild (oder Medium) ist. Bilder und Medien werden in dem Maße verkleinert, in dem das Elternelement schrumpft.
- Schrift verflüssigt man, indem man auf dem `<body>`-Element die Schriftgröße zurücksetzt und Schriftgrößen in Ems oder Prozent definiert.