

GRUNDLAGEN

# CASCADING STYLE SHEETS KOMPAKT



# WARUM STYLE SHEETS?

- Mit Stylesheets wird das Aussehen, sowie ein Teil des Verhaltens (Behaviour) eines HTML-Dokuments festgelegt.
- Damit bleibt das HTML Dokument als semantisches Struktur- und Inhaltedokument frei von Informationen, die nicht zu Struktur und Inhalt gehören.
- Eine Schriftgröße 24pt ist kein Content!

WEB OF DEVICES - BILDSCHIRM UND DRUCKER, SMARTPHONE, HEIZUNG UND KÜHLSCHRANK, FERNSEHGERÄT, BUSHALTESTELLE - ALLE HABEN INTERNETANSCHLUß.

- Durch die Trennung von Design und Content, kann das Aussehen des Contents durch das Verwenden mehrerer Stylesheets an das jeweilige Ausgabegerät und für die jeweilige Leserschaft angepasst werden.
- Die besteht übrigens aus Menschen und Maschinen, wenn Sie sich erinnern. CSS wird idR für Menschen geschrieben, obwohl es auch hier Ausnahmen gibt.

# CASCADING?

Eine Kaskade beschreibt eine Treppenstruktur.

Stylesheets können in verschiedenen Beziehungen zum Dokument positioniert werden. Damit sie sich dabei nicht gegenseitig im Weg stehen, zum Beispiel bei Mehrfachdefinitionen, folgen sie einer Abwärts- und Bedeutungshierarchie, Cascade genannt.

Aber: die Gewichtung eines Styles wird nach 'Spezifität' ermittelt. So kann eine hohe Spezifität die Cascade überbieten!

Siehe Beispiel unter [de.selfhtml.org](http://de.selfhtml.org) ...

# CASCADING STYLE SHEETS SYNTAX UND EINBINDUNG

# CASCADING STYLE SHEETS

```
body {  
    background-color : rgb(0, 0, 200);  
}
```

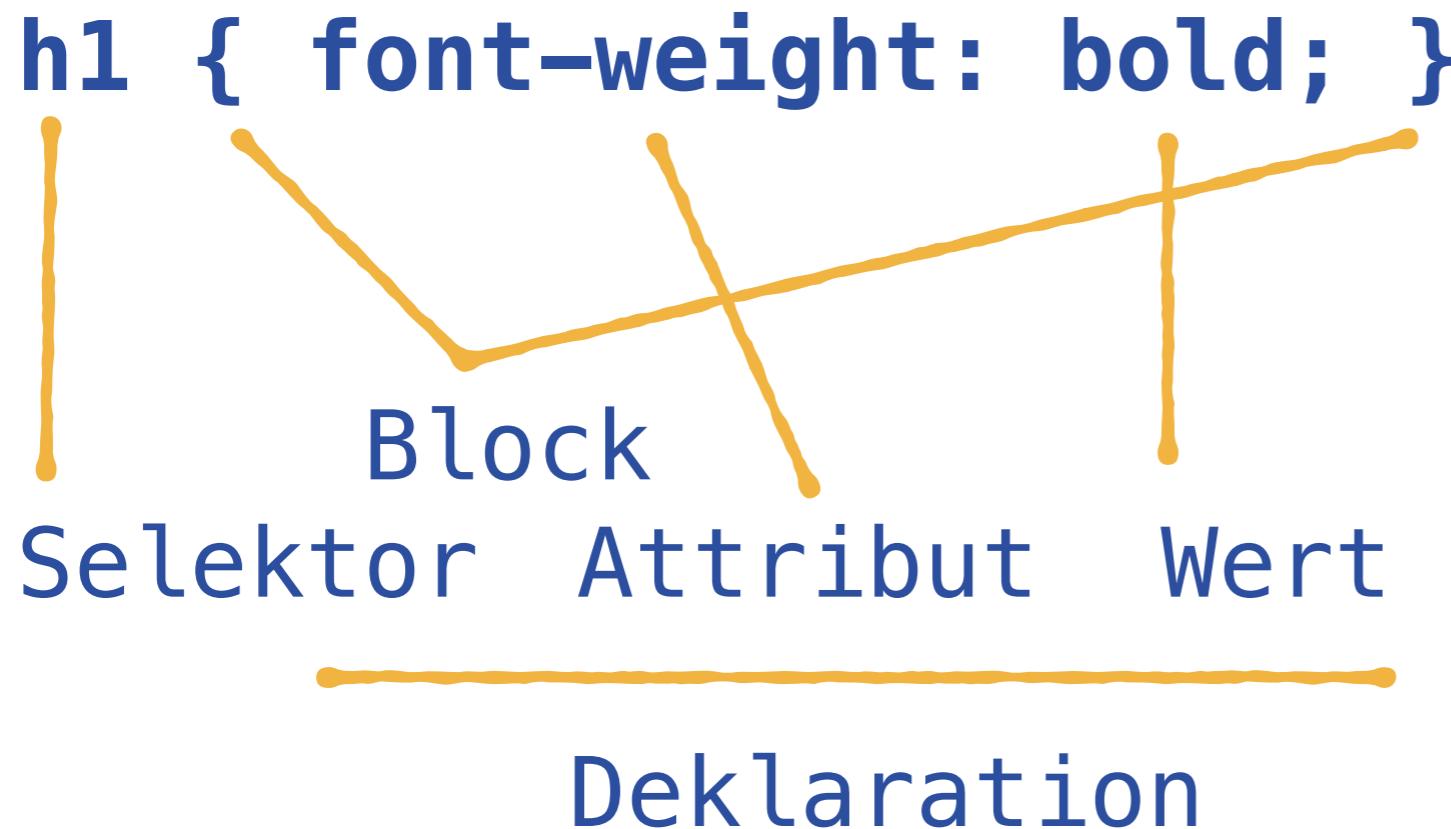
# STYLES ÄNDERN JEDE BELIEBIGE EIGENSCHAFT VON HTML-ELEMENTEN

```
background-color : rgba(0,0,0,0.15);  
background-image : url(meinbild.jpg);  
background-repeat : no-repeat;  
background-attachment : fixed;  
background-position : 10px 10px;
```

# CSS EIGENSCHAFTEN, BEISPIELSWEISE FÜR TEXT

color	→ Textfarbe
direction	→ Schreibrichtung
line-height	→ Zeilenabstand
letter-spacing	→ Buchstabenabstand
text-align	→ Textausrichtung
text-decoration	→ Textlinie
text-indent	→ Einrückung
text-shadow	→ Schatten
text-transform	→ Kapitälchen
unicode-bidi	→ Bidirektionalität
vertical-align	→ Ausrichtung, senkr.
white-space	→ Leerzeichenanzeige
word-spacing	→ Wortabstand
font-family	→ Schriftart
font-size	→ Schriftgrösse
font-weight	→ Schriftgewicht

# CSS BEFEHLSAUFBAU



# WERTEANGABEN IN VERSCHIEDENEN MAßSYSTEMEN & WERTEBEREICHEN

display: inline;

font-family:  
    Arial, "Times New Roman";

font-size:  
    0 | 12pt | 10pc | 13px  
    1em | 1ex | 1rem  
    100%;  
    100vh; 100vw

1em <- font-size des Parent-  
        Elements, z. B. 16px  
body { font-size : 15px; }  
h1 { font-size : 2em; }

1rem <- font-size des <html> Elements

# ZUSAMMENFASSEN MEHRERER ANWEISUNGEN

```
h1 { font-weight: bold; }
h1 { font-size: 12pt; }
h1 { line-height: 14pt; }
h1 { font-family: Arial; }
h1 { font-style: normal; }
```

```
h1 {
    font-weight : bold;
    font-size   : 12pt;
    line-height : 14pt;
    font-family : Arial;
    font-style  : normal;
}
```

# KURZSCHREIBWEISEN

```
h1 {  
    font-family: Arial, Verdana, sans-serif;  
    font-size: 12pt;  
    line-height: 14pt;  
    color: #000000;  
}  
  
h1 {  
    font: Arial 12pt/14pt #000;  
}
```

# KURZSCHREIBWEISEN

```
h1 {  
    margin-top      : 1em;  
    margin-right    : 1em;  
    margin-bottom   : 1em;  
    margin-left     : 1em;  
}  
  
h1 {  
    margin : 1em; /* TRBL */  
}  
h1 {  
    margin : 1em 2em; /* TB RL */  
}  
h1 {  
    margin : 1em 0 0 2em; /* T R B L */  
}
```

# EINBINDUNG EINER EXTERNEN CSS-DATEI IN HTML DOKUMENTE

```
<link href="assets/css/styles.css"  
      rel="stylesheet"  
      type="text/css" />
```

```
// mehrfach verwendbar  
// Gerätespezifisch  
// Möglichkeit der Modularität  
// Standardeinbindung!
```

# INTERNER STYLECONTAINER

```
<style>
  h1 { font-family: Arial; }
</style>

// einfach, nicht mehrfach verwendbar
// Anwendungsspezifisch.
// Möglichkeit der punktuellen
// temporären Ergänzung.
// Einbindung durch dynamischen Nachladen!
// Kann wieder entfernt werden.
```

# INLINE STYLES

```
<nav style="display : block;">
    Navigation
</nav>

// Einfach und elementenbezogen verwendbar
// Anwendungsspezifisch.
// Möglichkeit der punktuellen/temporären Ergänzung.
// Einbindung durch dynamischen Nachladen!
// Kann wieder entfernt werden.
// Wird dynamisch und zeitlich begrenzt,
// zum Beispiel von jQuery, erzeugt.
```

## USER AGENT STYLESHEET

ZURÜCKSETZEN – reset.css, normalize.css

LAYOUT, RESPONSIVE, (BASE), MODULES, SKINS – Mehrere externe Stylesheets

AJAX – Dynamische temporäre interne Styleelemente

DOM – ELEMENTEBENE  
Durch Javascript dynamisch verwaltete Inlinestyles.

# KASKADIERENDE EXTERNE CSS-DATEIEN

```
<link href="lib/css/normalize.css"  
      rel="stylesheet"  
      type="text/css" />  
  
<link href="src/css/responsive.css"  
      rel="stylesheet"  
      type="text/css" />  
  
<link href="src/css/skin.css"  
      rel="stylesheet"  
      type="text/css" />
```

# ERGÄNZEN DURCH KASKADIERUNG

```
body { font-family : Arial;  
      font-size : 16px;  
}  
h1 { font-size : 1.5em; }  
p { font-size : 0.75em; }  
  
<body>  
  <h1>...</h1>  
  <p>...</p>  
</body>
```

# SELEKTOREN

# CSS SELEKTOREN

body {}	→ Elementeselektor
.my-class {}	→ Klassenselektor
#my-id {}	→ ID-Selektor
a[href]	→ Attributselektor (minified)
input[type=email]	→ Attributselektor
[data-my-attr="value"]	→ Attributselektor
section#data {}	<section id="data"></section>
div.bar {}	

# GENERELLER SELEKTOR

- \* Stimmt mit jedem Element überein.

Er besitzt eine Spezifität von 0 und wird daher von jeder anderen Styleanweisung überschrieben.

Eine Spezifität von 1 erreicht man mit

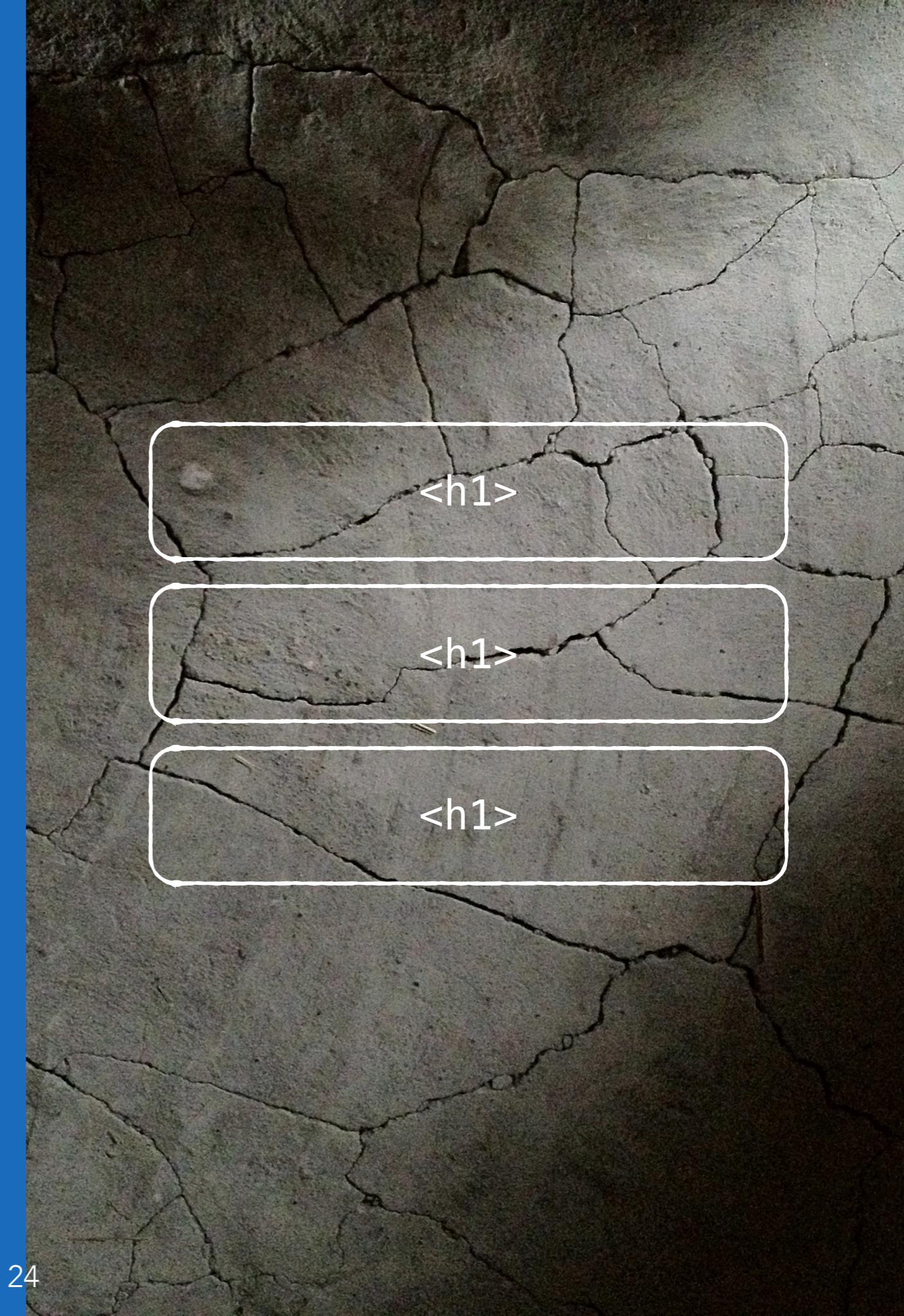
```
body * { ... }
```

# ELEMENT SELEKTOR

body

Stimmt mit jedem E-Element  
überein  
(d. h. ein Element des Typs  
E).

→ `h1 {}`

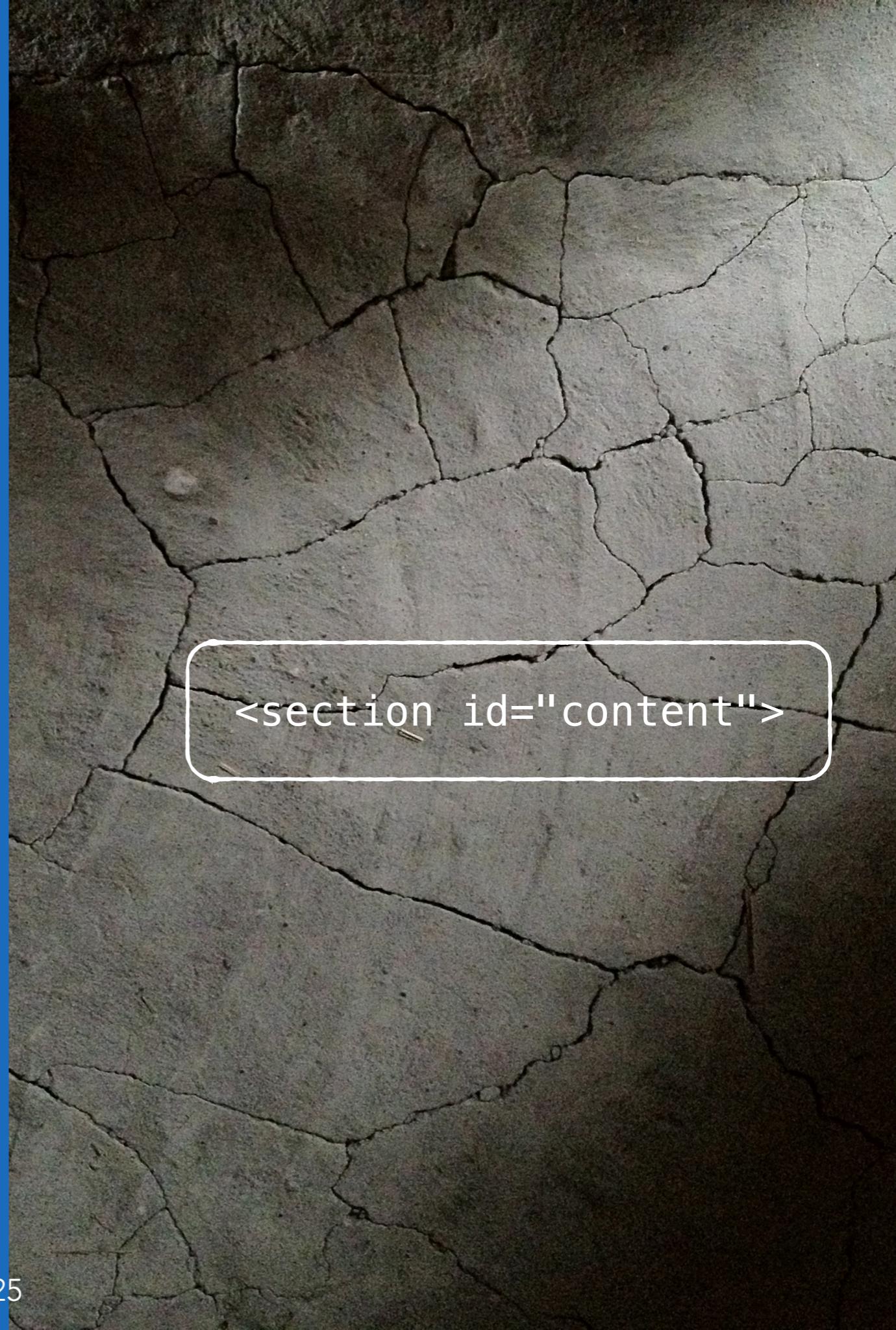


# ID SELEKTOR

#my-id

Eine ID muss im Dokument unique sein!

→ section#content {}  
→ #content {}



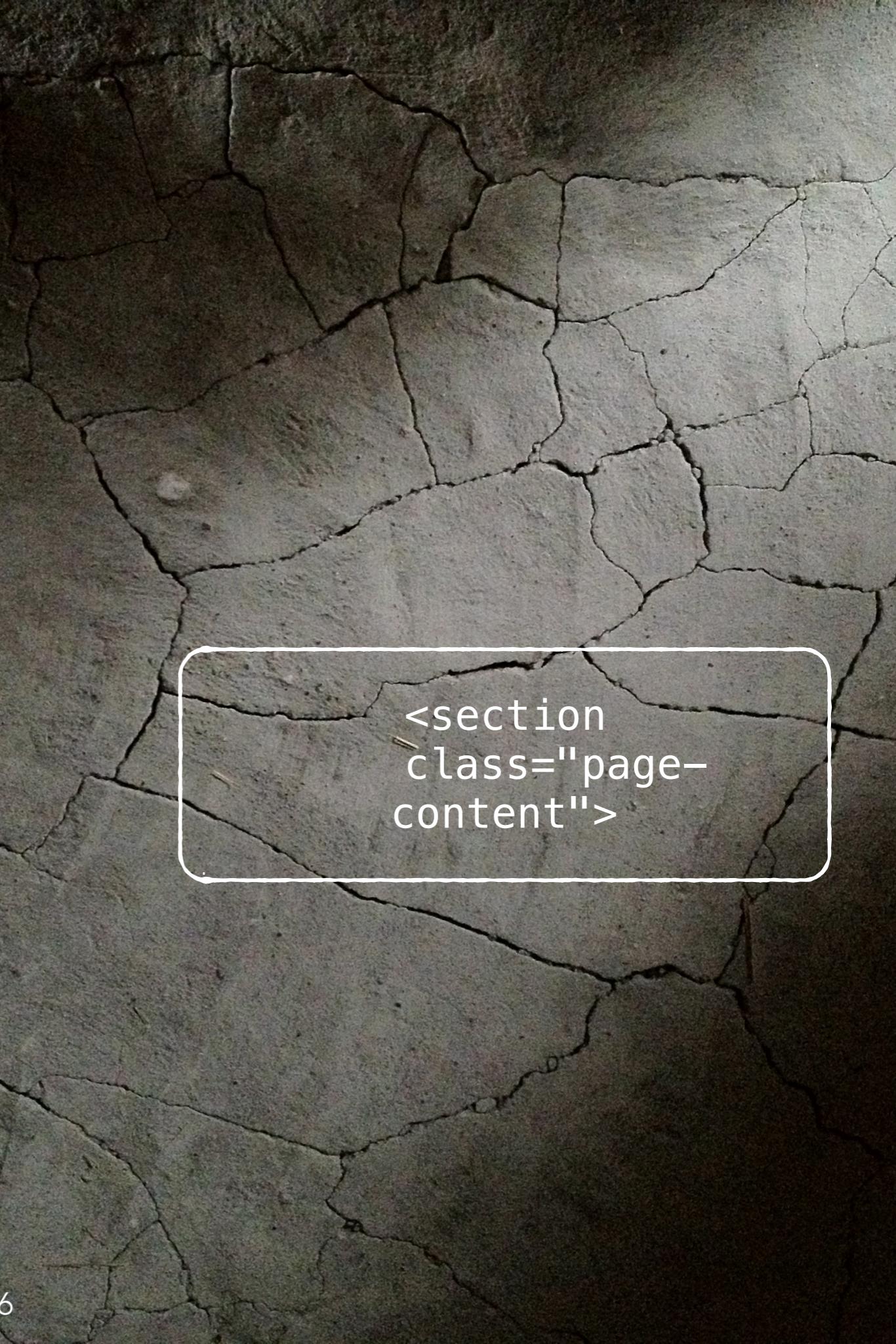
<section id="content">

# KLASSENSEL KTOR

.my-class-name

Klassen sind mehrfach  
verwendbar.

→ section.page-content {}  
→ .page-content {}



# ATTRIBUTSELEKTOREN

[foo]

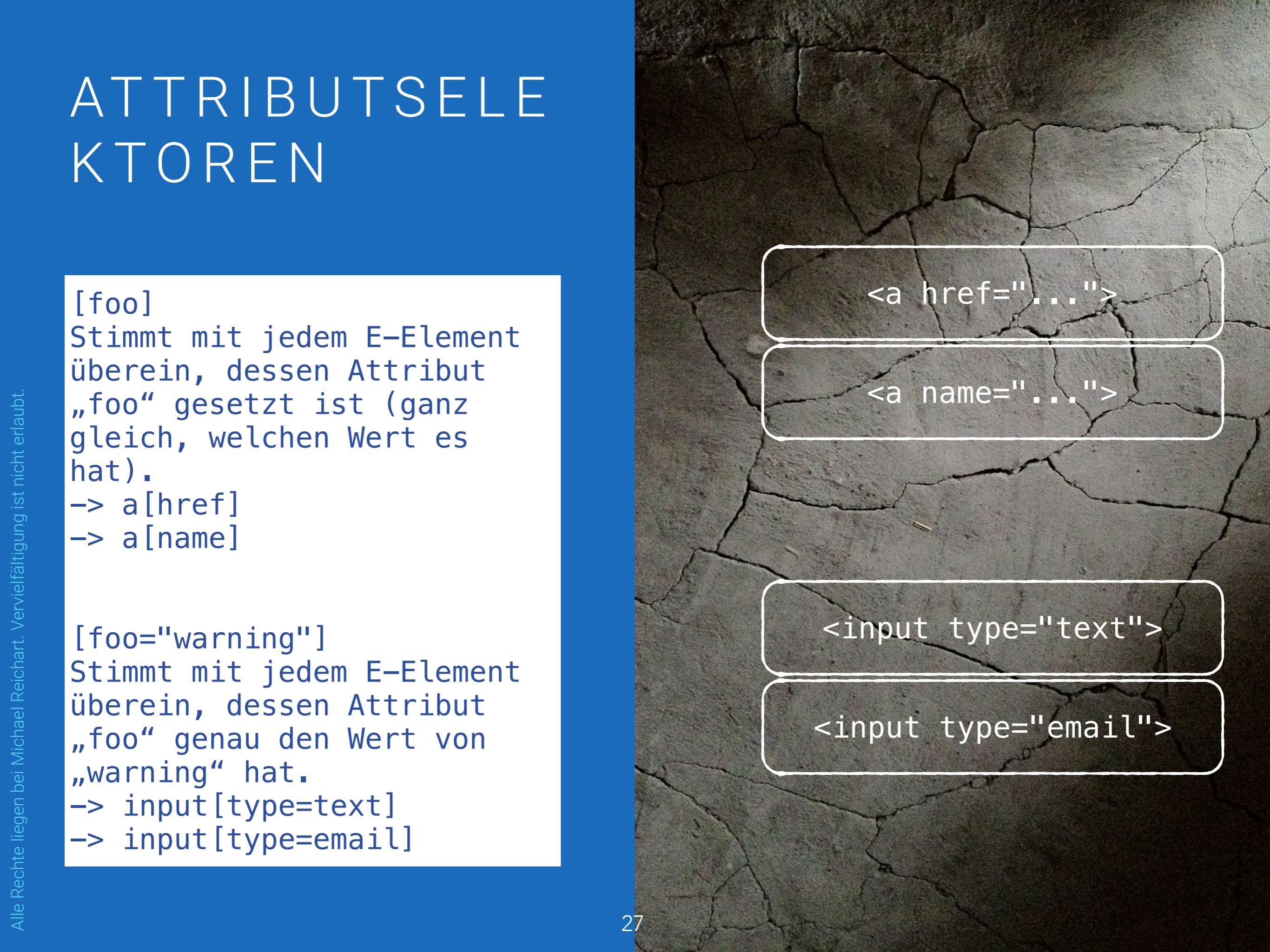
Stimmt mit jedem E-Element überein, dessen Attribut „foo“ gesetzt ist (ganz gleich, welchen Wert es hat).

- a[href]
- a[name]

[foo="warning"]

Stimmt mit jedem E-Element überein, dessen Attribut „foo“ genau den Wert von „warning“ hat.

- input[type=text]
- input[type=email]



<a href="...">

<a name="...">

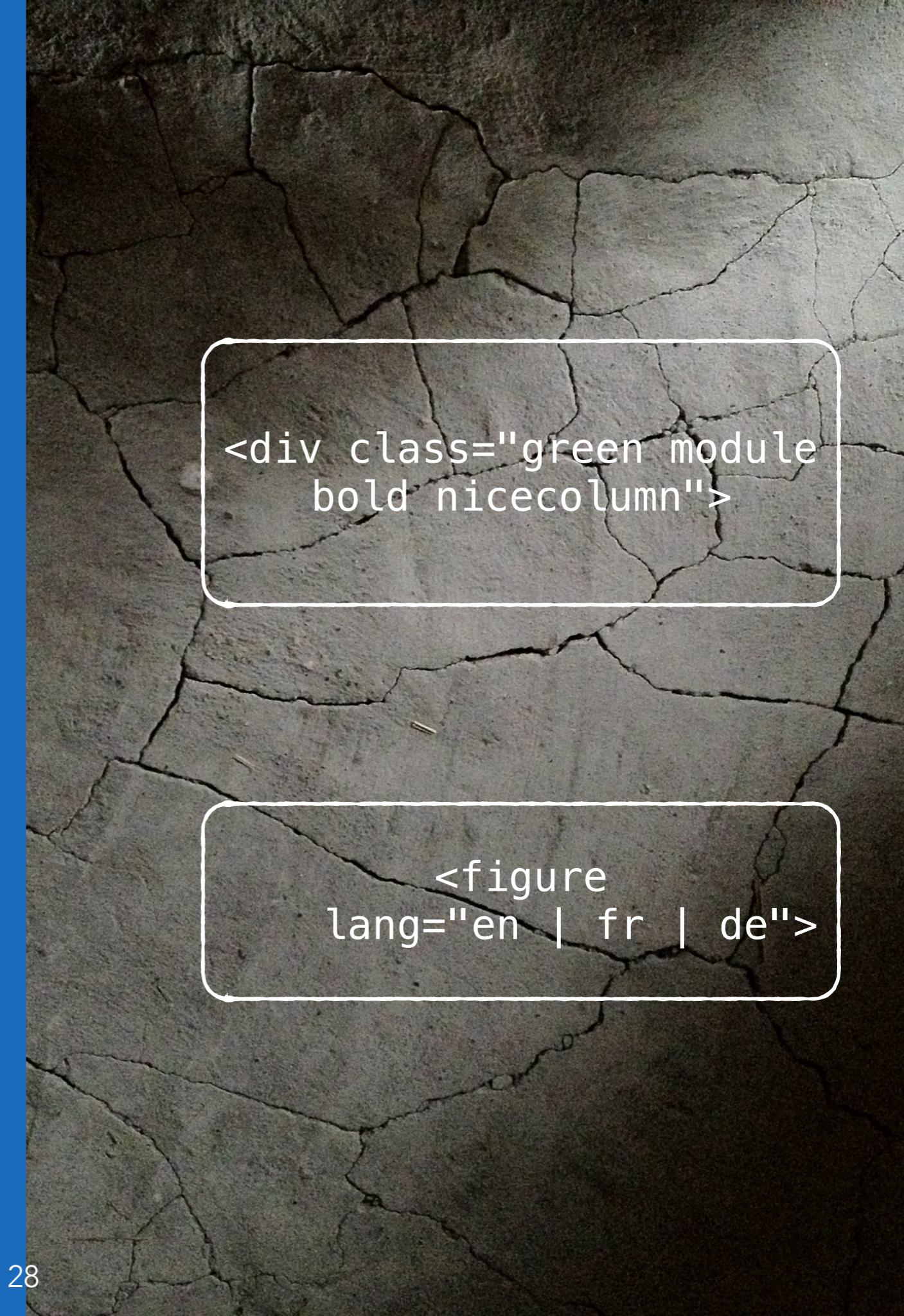
<input type="text">

<input type="email">

# ATTRIBUTSELEKTOREN

E[foo\*=bar]  
Liste von durch Kommas voneinander getrennten Werten, und einer dieser Werte ist gleich „bar“.  
-> div[class\*=column]

E[lang|=en]  
Liste mit durch Trennstriche voneinander getrennten Werten, die (von links) mit „en“ beginnen.



# VERNEINUNG VON ATTRIBUTEN

```
input[type=text] {  
  background: #eee;  
}  
input[type!=text] {  
  
a[href] { ... }  
a[!href]
```

# CSS SELEKTOREN KOMBINATIONEN

`div.page {}` → eingeschränkter Klassenselektor

`* {}` → General Selektor

`ol li` → Descendantselektor

`nav > ul > li` → Childselektor

`ul > li ~ li` → General Following Siblings

`h1 + p` → Immediate Following Sibling

`ul > li:first-child`

`h1 ~ p:first-child`

`ul li:nth-child(even)`

`ul li:nth-child(odd)`

# DESCENDANT SELECTOR

Alle Rechte liegen bei Michael Reichart. Vervielfältigung ist nicht erlaubt.

Das Leerzeichen zwischen  
zwei Selektoren

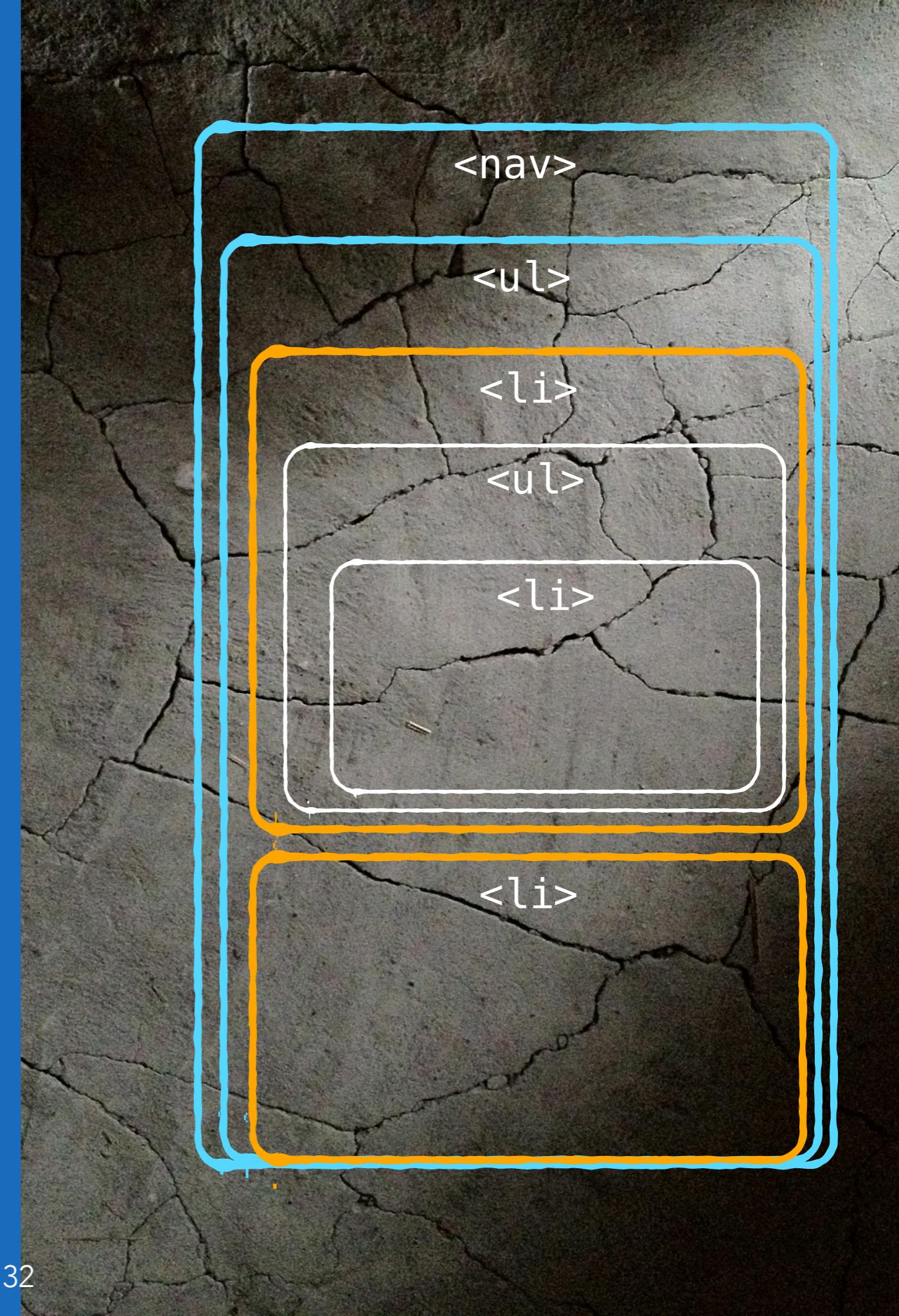
```
-> nav li { list-style :  
none; }
```



## > CHILD SELECTOR

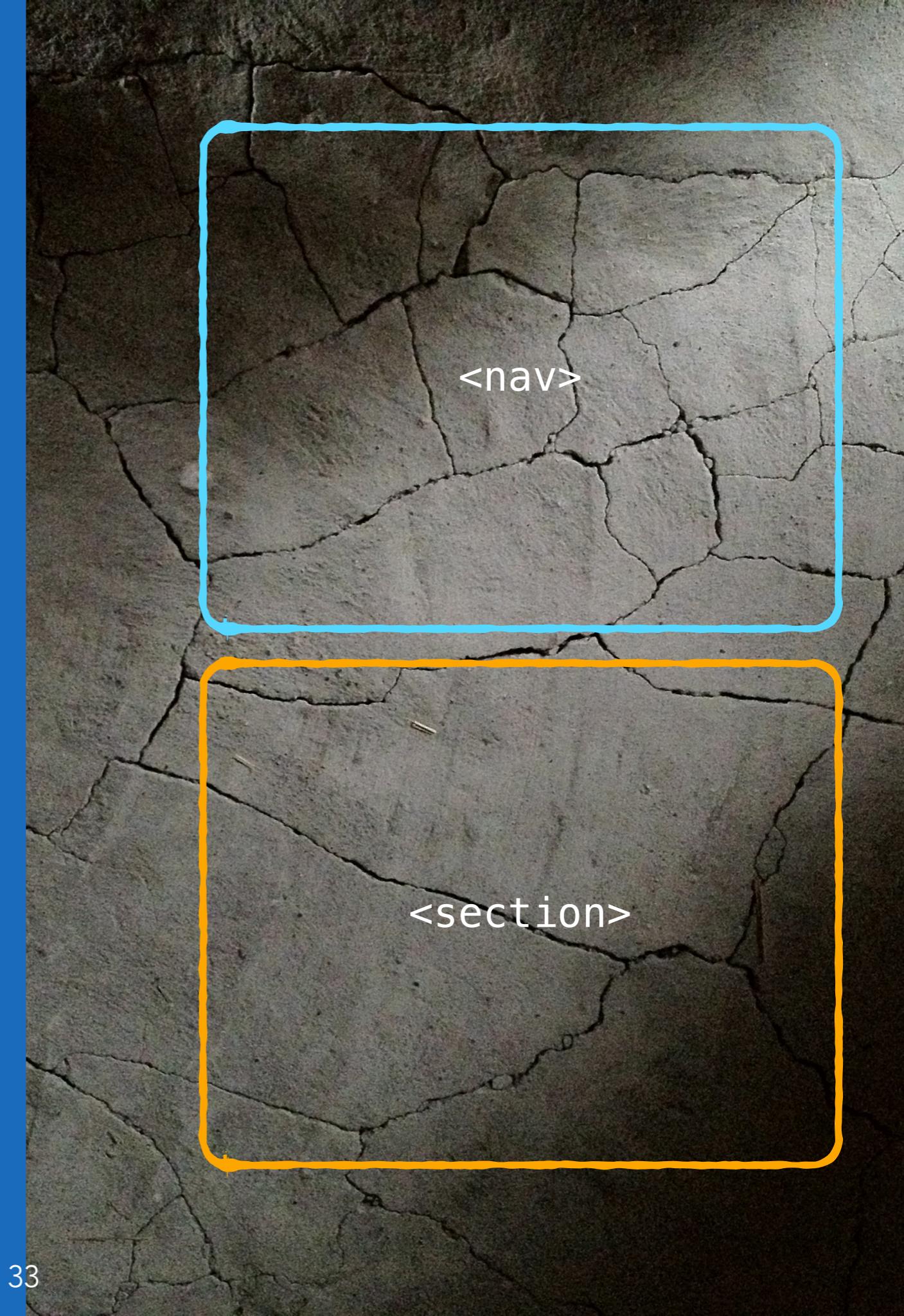
Stimmt mit allen F-Elementen überein, die Kindelemente eines Elements E sind.

→ nav > ul > li {}



+  
ADJACENT SIBLING  
(IMMEDIATE  
FOLLOWER)

Stimmt mit jedem F-Element  
überein, dem unmittelbar  
ein Element E vorausgeht.  
→ nav + section {}

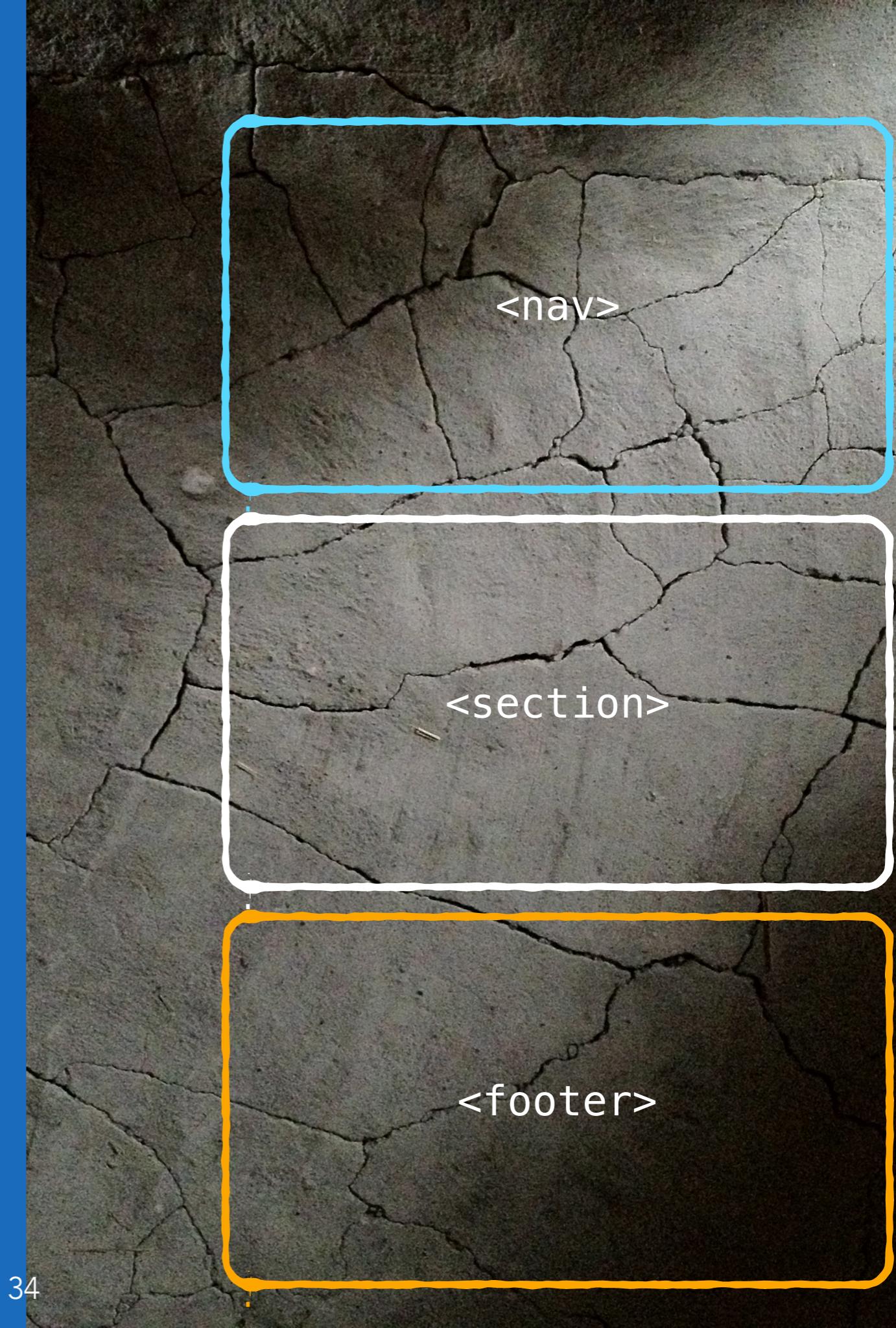


~

## GENERAL FOLLOWING SIBLING

Stimmt mit jedem folgenden  
Geschwisterelement überein.

-> nav ~ footer {}



# KOMPLEXE CSS SELEKTOREN

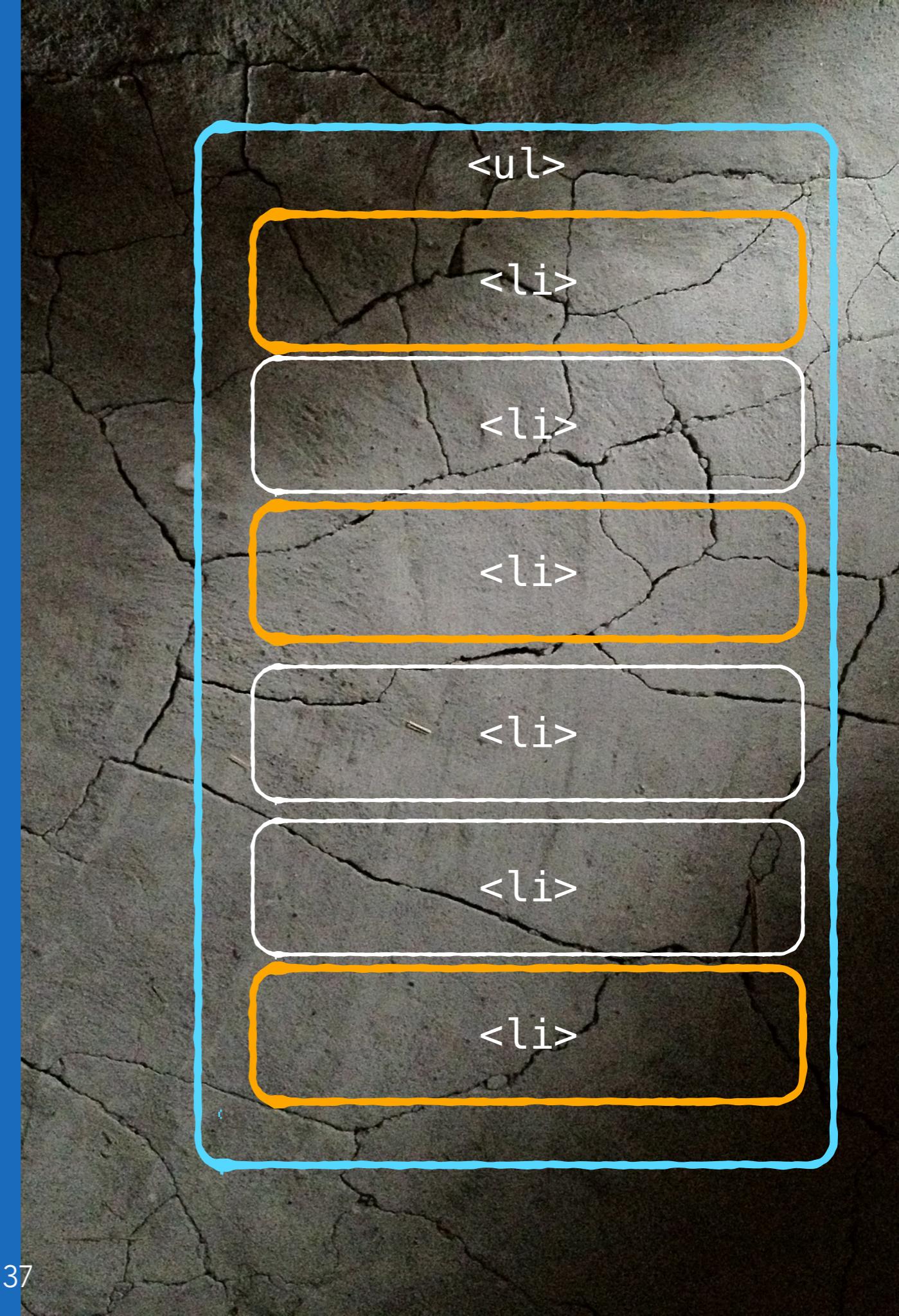
```
body * {}
body [class^=page-]
nav > ul > li
[class=page-header], [class=page-header] ~ [class*=page-]
[class*=column] + [class*=column]:last-child
```

# CSS SELEKTOREN PSEUDOKLASSEN

```
a[href]:hover          -> Stateselector (Pseudoklasse)
[class*=page-]::before {} -> Pseudoelement
nav li:last-child {}    -> Pseudoklasse
                           -> Sprachattribut Selektor
:lang(de-DE)
```

# PSEUDO-KLASSEN

```
ul li:first-child  
ul li:nth-child(3)  
ul li:last-child
```



# NTH-CHILD

```
h2:first-child { ... }

h2:last-child { ... }

.row:nth-child(even) { /*Gerade Zeilen */
  background: #dde;
}

.row:nth-child(odd) { /* Ungeraden Zeilen */
  background: white;
}

.row:nth-child(5n) { /* Jede fünfte Zeile */
  background: #dde;
}

.row:nth-child(5n+1) { /* Jede fünfte Zeile */
  background: #dde;
}
```

# NTH-OF-TYPE()

```
div#test p:nth-of-type(4) { ... }  
div#test p:first-of-type { ... }  
div#test p:last-of-type { ... }  
div#test p:only-of-type { ... }
```

# NTH-CHILD() VS. NTH-OF-TYPE

```
<style>
  footer a:nth-child(odd) { color : lightblue; }
  footer a:nth-of-type(odd) { color: green; }
</style>
```

# NTH-CHILD() VS. NTH-OF-TYPE

```
<footer>
  <a href="">link</a>
  <span>Span</span>
  <a href="">link</a>
  <a href="">link</a>
  <span>Span</span>
  <span>Span</span>
  <a href="">link</a>
  <a href="">link</a>
  <a href="">link</a>
  <a href="">link</a>
</footer>
```

```
<footer>
  <a href="">link</a>
  <span>Span</span>
  <a href="">link</a>
  <a href="">link</a>
  <span>Span</span>
  <span>Span</span>
  <a href="">link</a>
  <a href="">link</a>
  <a href="">link</a>
  <a href="">link</a>
</footer>
```

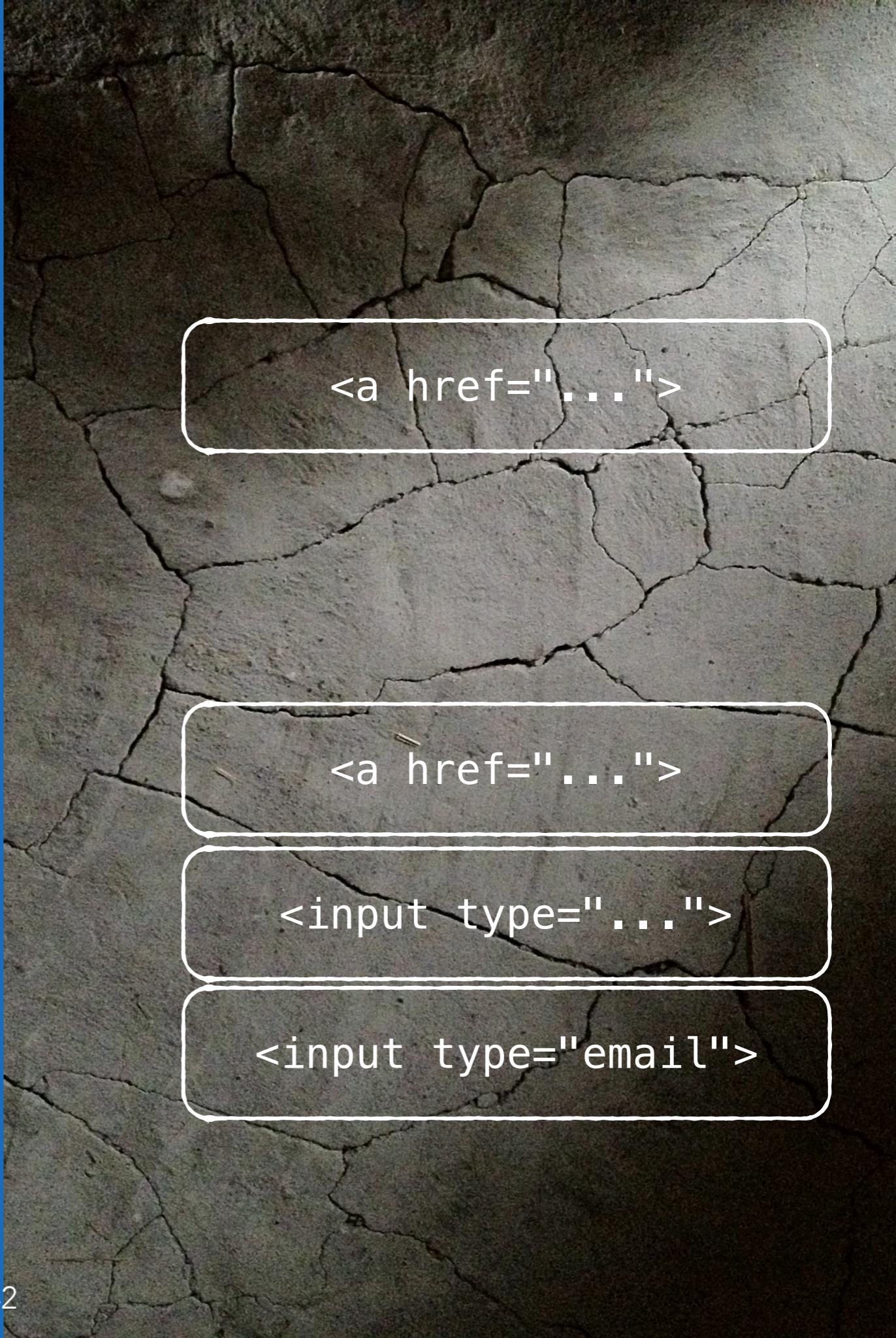
# ZUSTANDS- SELEKTOREN

`a:link, a:visited`

Stimmt mit dem Anchor überein, dessen Ziel noch nicht besucht wurde (`:link`), oder dessen Ziel bereits besucht wurde (`:visited`).

`a:active, a:hover,`  
`input:focus`

`input:valid, input:invalid`  
Reflektiert ein Validierungsergebnis auf feldtypenspezifische Eingaben.



<a href="....">

<a href="....">

<input type="....">

<input type="email">

# SPRACH- PSEUDO-KLASSE

:lang(c)  
→ p:lang(fr)



# :: FÜR PSEUDOELEMENTE

```
<span data-currency=„ €“>10</span>

span[data-currency]::after {
    content : attr(data-currency);
}

<span data-currency=„ €“>10 €</span>
```

# TOOLTIIPP PER PSEUDOELEMENT

```
<span data-desc="tooltip text content">?</span> Text

span[data-descr] {
    position : relative;
    color    : #00F;
    cursor   : help;
    text-decoration : underline;
}

span[data-descr]:hover::after {
    content      : attr(data-descr);
    position     : absolute;
    left         : 0;
    top          : 24px;
    min-width   : 200px;
    background-color : #fffffcc;
    padding      : 12px;
    color        : #000000;
    font-size    : 14px;
    z-index      : 1;
    border       : 1px #aaaaaa solid;
    border-radius : 10px;
}
```

# NEGATION VON SELEKTOREN

```
:not(.box) {  
    color: #00c;  
}  
:not(span) {  
    display: block;  
}  
  
h2:not(.teaser)  
statt h2.default, h2.special  
<h2 class="teaser"></h2> -> wird nicht ausgewählt!  
<h2 class="default"></h2>  
<h2 class="special"></h2>
```

# GEWICHTUNGEN

# CSS GEWICHTUNGEN

*	0	0	0	0	0
body	0	0	0	0	1
.my-class	0	0	0	1	0
:pseudo	0	0	0	1	0
[href]	0	0	0	1	0
#id	0	0	1	0	0
<b style="">	0	1	0	0	0
!important	1	0	0	0	0

# CSS GEWICHTUNGEN

nav > ul > li	0 0 0 0 3
a[href]	0 0 0 1 1
#form-login	0 0 1 0 0
a[href] { ... !important; }	1 0 0 1 1

# CSS GEWICHTUNGEN

```
<body class=„red“> ... </body>

.red {
    background-color : red;
}

body {
    background-color : black !important;
}
```

# SKALIERBARES, MODULARES CSS

# STYLESHEETS STRUKTURIERT AUFBAUEN

- CSS können jedes Element und jede Eigenschaft einer Weboberfläche ansprechen und gestalten.
- - ist eine deklarative Sprache
- - keine Struktur-Befehle
- - die Struktur muss konzeptionell erreicht werden.

# KATEGORISIEREN!

- Die Aufgaben von Stylesheets können in immer gleiche Kategorien eingeteilt werden:
  - - Grundsätzliches, Semantisches
  - - Layout und Anordnung
  - - Zustandsveränderungen
  - - Projekt oder Kundenanpassungen

# KATEGORISIEREN!

**base.css**

**layout.css**

**module-1.css**

**module-2.css**

**theme.css**

# DER INTERNE AUFBAU EINER DEKLARATION

**/\* BASE \*/**

Alles, was grundsätzlich in einem Modul oder einem Element gelten soll.

**/\* LAYOUT \*/**

Anordnungsvarianten, zum Beispiel vertikale oder horizontale Anordnungen.

**/\* STATES \*/**

Zustandveränderungen bei Mausbewegung, aktivem Cursor et.

# BASE.CSS

```
html,  
html * { box-sizing : border-box; }  
  
/* FARBEN */  
/* Hue, Saturation, Lightness (Luminance) */  
html {  
    background-color : hsla(360, 0%, 80%, 1);  
    color           : hsla(180, 0%, 20%, 1);  
}  
a[href] {  
    color : hsla(210, 70%, 30%, 1);  
}  
  
/* TYPOGRAPHY */  
html {  
    font-family : sans-serif;  
    font-size   : 14px;  
    line-height : 1.428571; // entspricht 20px  
}  
p,  
ul,ol,li,  
td,th,  
code { font-size : 1.0rem; margin-top: 1rem; margin-bottom: 1rem; }  
h1 { font-size : 2.0rem; margin-top: 1rem; margin-bottom: 1rem; }  
h2 { font-size : 1.8rem; margin-top: 1rem; margin-bottom: 1rem; }  
h3 { font-size : 1.6rem; margin-top: 1rem; margin-bottom: 1rem; }  
h4 { font-size : 1.4rem; margin-top: 1rem; margin-bottom: 1rem; }  
h5 { font-size : 1.2rem; margin-top: 1rem; margin-bottom: 1rem; }  
h6 { font-size : 1.0rem; margin-top: 1rem; margin-bottom: 1rem; }
```

# TEMPLATE.CSS

```
body {  
    display : flex;  
    justify-content : space-around;  
}  
  
.page {  
    display : block;  
    width : 960px; }  
.page-header {  
    display : block;  
    width : 100%; }  
.page-content {  
    display : flex;  
    width : 100%;  
    flex-direction : row;  
    flex-wrap : wrap;  
    justify-content : center;  
    align-items : stretch;  
    align-content : flex-start;  
}  
.page-footer {  
    display : block;  
    width : 100%; }  
  
.content-nav {  
    display : block;  
    order : 1;  
    flex-basis : 25%;  
    flex-grow : 1;  
    flex-shrink : 0;  
}  
.content-main {  
    display : block;  
    order : 2;  
    flex-basis : 50%;  
    flex-grow : 1;  
    flex-shrink : 0;  
}  
.content-aside {  
    display : block;  
    order : 3;  
    flex-basis : 25%;  
    flex-grow : 1;  
    flex-shrink : 0;  
}
```

# NAVIGATION.CSS

```
/* BASE */
.nav {
  list-style : none;
  margin     : 0;
  padding    : 0;
}

.nav h4 {
  font-size  : 1em;
  font-weight: 300;
}

.nav li a[href] {
  display : inline-block;
  width   : 100%;
  height  : 100%;
}

.nav {
  margin-left   : -1rem;
  margin-right  : -1rem;
}
.nav li {
  margin-top    : 0;
                                margin-bottom : 0;
}

/* LAYOUT */
.nav-vertical li  {
  display : block;
}
.nav-horizontal li {
  display : inline-block;
}
.nav-left li    {
  display : block;
  float   : left;
}
.nav-left li::after {
  content : "";
  display : table;
  clear   : left;
}
.nav-right li   {
  display : block;
  float   : right;
}
```

<http://smacss.com>

Hier hat Jonathan Snook Regeln und Beispiele für eine skalierbare und modulare Vorgehensweise beschrieben und mit Beispielen anschaulich illustriert.

# GENERISCHES CSS

# STYLESHEETS GENERISCH

- Mit SASS (Syntactically Awesome Stylesheets) können Stylesheets "programmiert" werden.
- Variablen halten wiederverwendbare Werte für Farben, Schriften, Breiten und mehr.
- "Mixins" beinhalten wiederverwendbare Deklarationen und aufrufbare Funktionen.
- Ein Compiler generiert aus den SASS-Codes CSS-Zeilen und legt diese in einem Verzeichnis ab.

# \_VARIABLES.SCSS

```
// LAYOUT
$default-padding: 0.5rem;
$default-margin: 0;

// FONTS
$font-size: 22px;
$line-height: 1.8rem;

// COLORS
$text-hue: 0;
$text-saturation: 0;
$text-lightness: 10%;
$text-alpha: 1;
$color-hue: 210;
$color-saturation: 80%;
$color-lightness: 50%;
$color-alpha: 1;
```

# \_MIXINS.SCSS

```
@mixin border-radius($radius) {  
  -webkit-border-radius: $radius;  
  -moz-border-radius: $radius;  
  -ms-border-radius: $radius;  
  border-radius: $radius;  
}  
  
.box { @include border-radius(10px); }
```

# IMPORTS

```
// _reset.scss
html, body, ul, ol {
  margin: 0;
  padding: 0;
}

// base.scss
@import 'reset';

body {
  font: 100% Helvetica, sans-serif;
  background-color: #efefef;
}
```

# MIT EINRÜCKUNGEN ARBEITEN

```
nav {  
    ul {  
        margin: 0;  
        padding: 0;  
        list-style: none;  
    }  
  
    li {display:inline-block;}  
  
    a {  
        display: block;  
        padding: 6px 12px;  
        text-decoration: none;  
    }  
}
```

```
nav ul {  
    margin: 0;  
    padding: 0;  
    list-style: none;  
}  
  
nav li {  
    display: inline-block;  
}  
  
nav a {  
    display: block;  
    padding: 6px 12px;  
    text-decoration: none;  
}
```

```
$ cd path/to/web/directory  
$ sass --watch assets/scss:assets/css  
  
--style=expanded | compact | compressed | nested  
--sourcemap=None
```

# GESTALTUNG VON SEITENVORLAGEN TEMPLATES

# DISPLAY, WIDTH, HEIGHT, PADDING, MARGIN DAS BOX MODEL

DIE GRÖSSE VON HTML ELEMENTEN BERECHNEN

# DAS BOX MODEL

margin-left

margin-right

padding-left

padding-right

Etiam porta sem  
malesuada magna  
mollis euismod.  
(border-)width

Html Tricks  
komplette Seite  
bauen

komplette Seite  
bauen mit  
Menüleiste oben

Flexbox,  
Gridlayout

praktische  
Anwendung –  
Webseite nachbilden

# BOX MODEL ADDITION

## **box-sizing: content-box**

Dimensionsverhalten wie in CSS 2 – Höhe, Breite werden definiert, padding, border, margin und outline werden dazu gerechnet.

## **box-sizing: border-box**

In die Boxbreiten- und Höhenangabe werden padding und border eingerechnet.

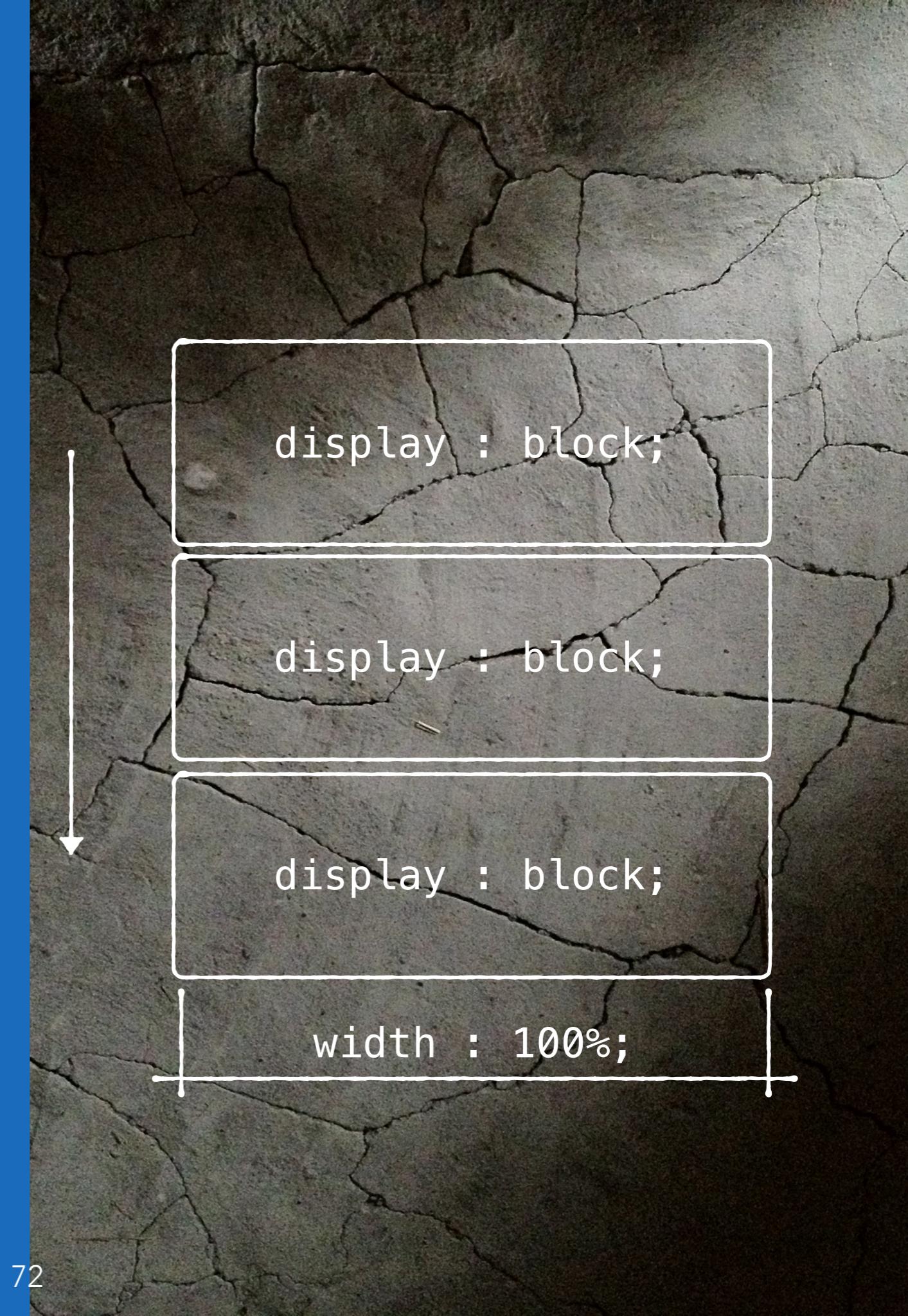
```
div.row-fluid {  
    box-sizing : border-box;  
}
```

## **box-sizing: padding-box**

Padding mit in die width Dimension

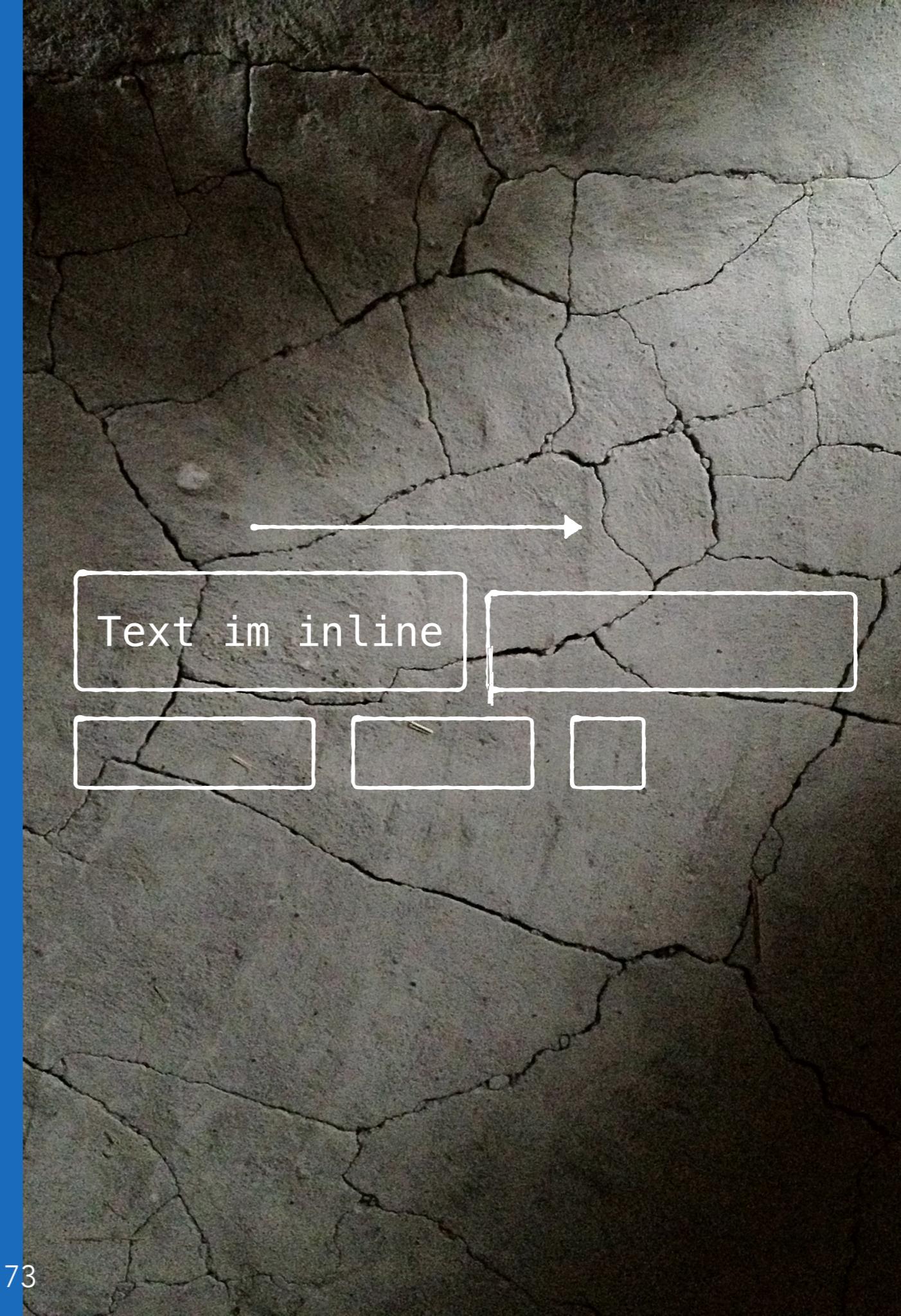
# BLOCK

```
div {  
  display : block;  
  width   : 100%;  
  [height  : auto;]  
  margin   : 1em;  
  padding  : 1em;  
}
```



# INLINE

```
div {  
    display : inline;  
    margin  : 1em;  
    padding : 1em;  
}
```

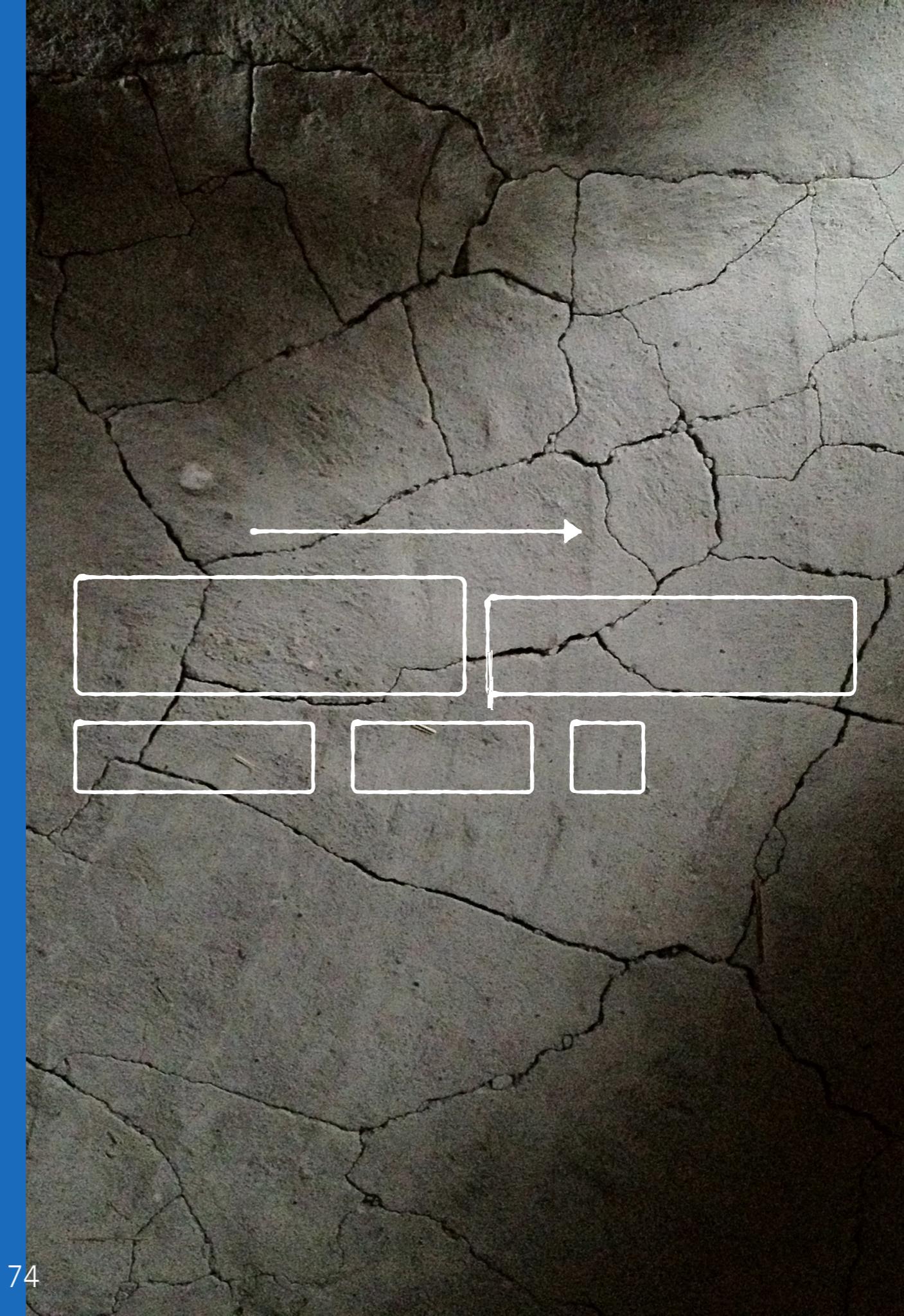


# INLINE-BLOCK

```
div {  
  display: inline-block;  
}
```

Stattet das Inlineelement mit width/height aus, auf ein float Attribut kann hier teilweise verzichtet werden.

```
div {  
  display : inline-block;  
  width   : 12em;  
  height  : 12em;  
  margin  : 1em;  
  padding : 2em;  
}
```



# ELEMENT MIT TABELLENVERHALTEN

```
div {  
  display: table;  
}  
div {  
  display: table-cell;  
}
```

Ein Blockelement erhält eine Höhe und verdrängt damit Elemente, auch wenn nichts drin steht.  
Lässt eine vertikale Zentrierung zu!

# MICRO CLEARFIX VON GALLAGHER

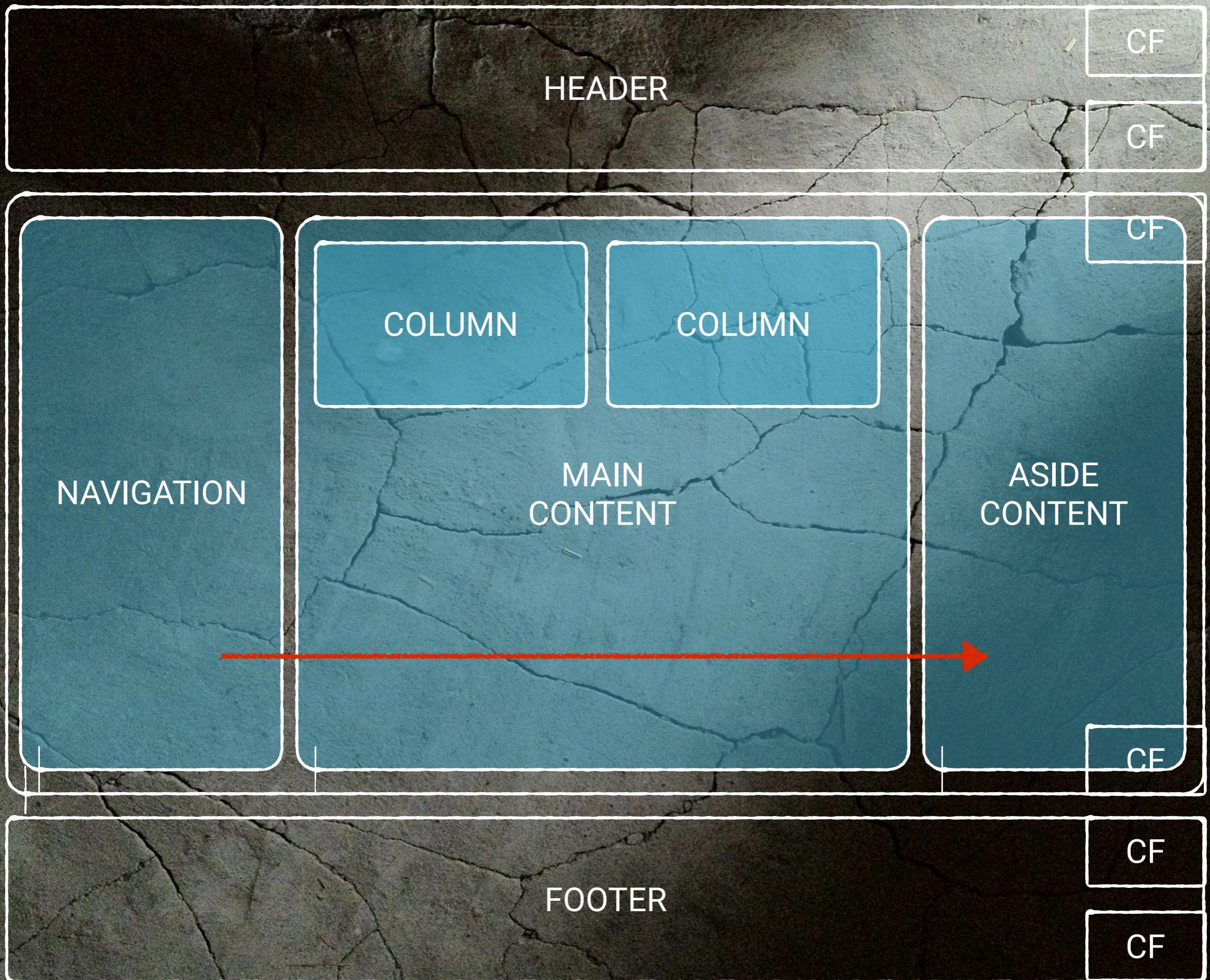
```
.cf:before, .cf:after {  
    content: " ";  
    display: table;  
}  
.cf:after {  
    clear: both;  
}  
.cf {  
    *zoom: 1; /* for IE 6/7 only */  
}  
  
/* seit CSS3: element::after { ... } */
```

# NEUES LAYOUTEN FLOATING LAYOUTS

HEADER

CONTENT

FOOTER



# DAS HTML GERÜST

```
<div class="page" id="news">  
  <div class="page-header"> . . . </div>  
  
  <div class="page-content">  
    <div class="content-nav"> . . . </div>  
    <div class="content-main"> . . . </div>  
    <div class="content-aside"> . . . </div>  
  </div>  
  
  <div class="page-footer"> . . . </div>  
</div>
```

# FLOAT UND CLEAR

```
.page-header { display : block; width : 100%; clear : both; }
.page-content { display : block; width : 100%; clear : both; }
.page-footer { display : block; width : 100%; clear : both; }

.content-nav { display : block; width : 25%; float : left; }
.content-main { display : block; width : 50%; float : left; }
.content-aside{ display : block; width : 25%; float : left; }
```

# MICRO CLEARFIX VON GALLAGHER

```
.page-header::before, .page-header::after,  
.cf:before, .cf:after {  
    content: " ";  
    display: table;  
}  
.page-header::before, .page-header::after,  
.cf:after {  
    clear: both;  
}  
.cf {  
    *zoom: 1; /* for IE 6/7 only */  
}  
  
/* seit CSS3: element::after { ... } */
```

# HILFREICH BEIM GESTALTEN UND UMSETZEN VON SEITEN. SEITENRASTER

# GESTALTUNGSRASTER

- Gestaltungs raster oder Grids sind ein Mittel, grafische Elemente mit Texte und Bilder so im Arbeitsbereich einzuordnen, dass Übersichtlichkeit und Struktur entsteht.
- Der Satzspiegel ist das älteste und bekannteste Gestaltungs raster aus der Typografie, das schon aus der mittelalterlichen Buchkunst bekannt ist.

# UMSETZUNG EINES GRIDS MIT HTML UND CSS

- Grids bestehen aus modularen Stylesheets, die meist auf Seiten- und Inhaltsebene organisiert sind.
- Ihre Anwendung verwendet eine zum großen Teil festgelegte HTML Struktur, sowie vordefinierte Klassen (oder id-) Attribute.

# BEKANNTES GRIDSYSTEME

- Yaml (Yet Another Multicolumn Layout)
- jQuery Mobile (Für mobile Geräte)
- 960 (der Klassiker für Standardbildschirme)
- Bootstrap (Twitter Prototyping Tool)
- Foundation

# FESTE UND FLEXIBLE BREITEN

- Ein Layout kann so gestaltet werden, dass es immer eine feste Breite hat oder sich dem Browserfenster und den Aktionen des Anwenders anpasst.
- Mit der zunehmenden Bedeutung mobiler Geräte spielen flexible Layouts, insbesondere responsive Layouts eine wichtige Rolle.

60 px															
-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

10 Spalten

3 Spalten

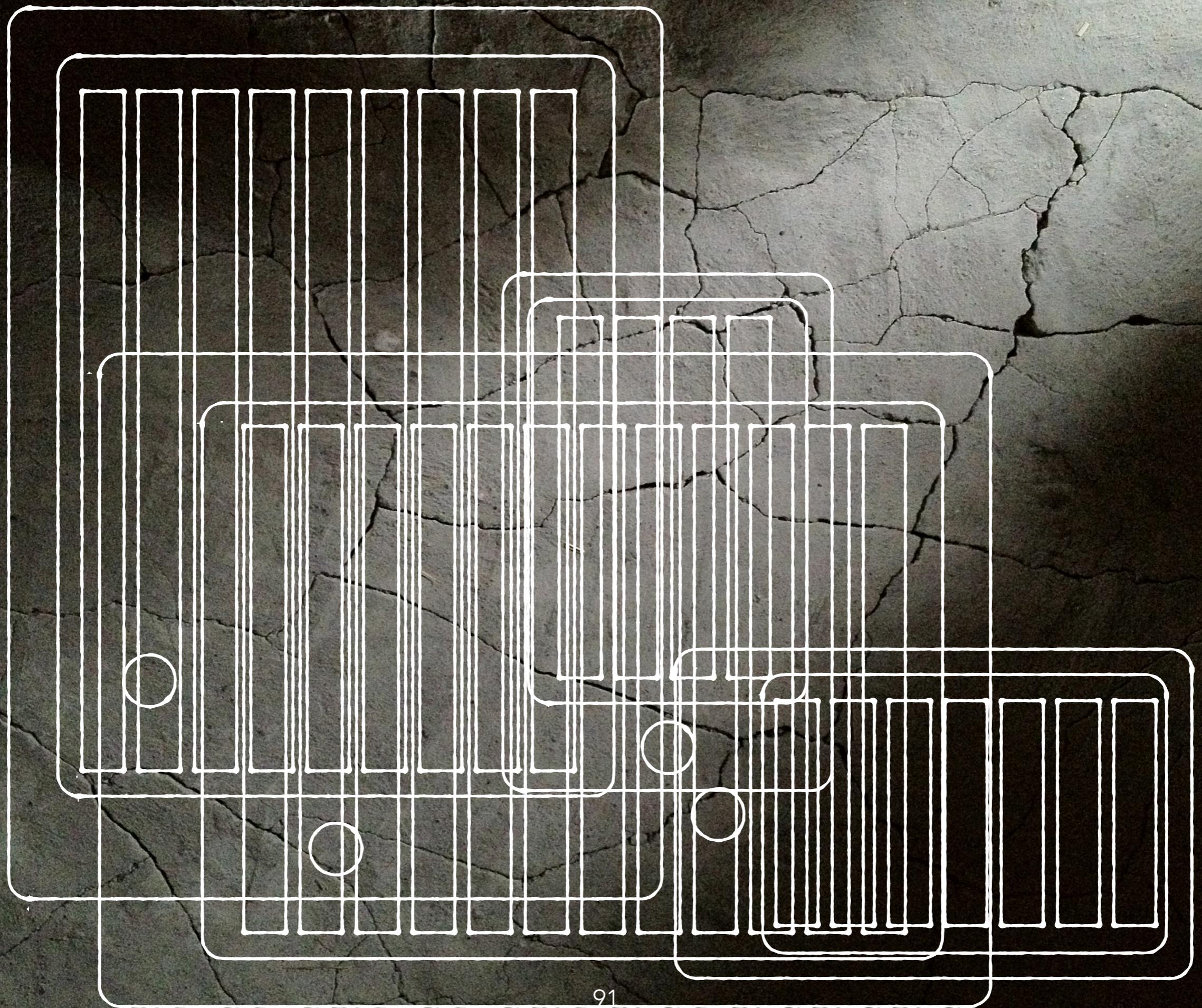
5 Spalten

5 Spalten

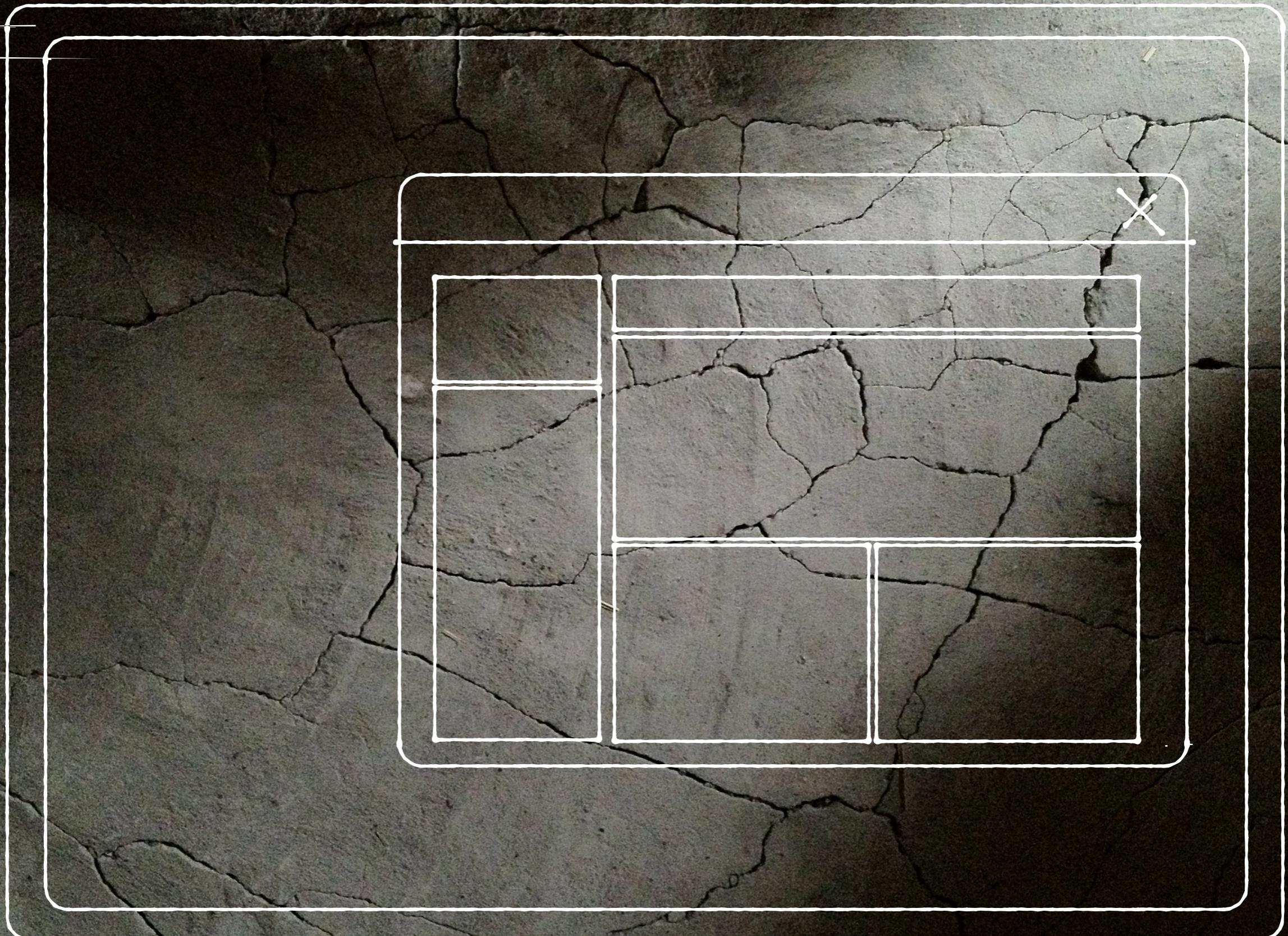
3 Spalten

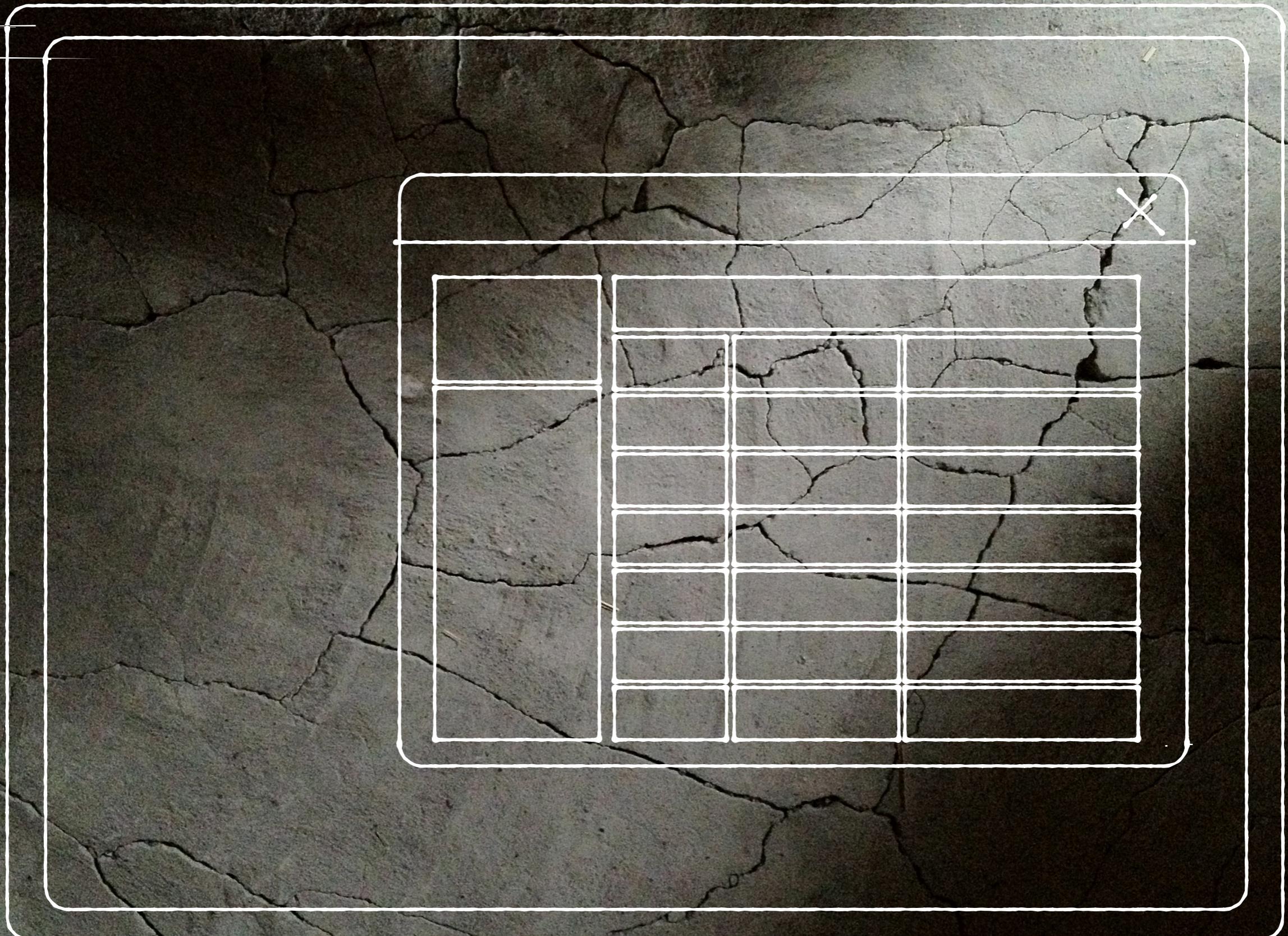












# EIN GRID SCHREIBEN.

# WAS WIR VON EINEM GRID ERWARTEN ...

- Flexibilität und Anwendbarkeit
- Mehrspaltigkeit. Das folgende Beispiel arbeitet mit einem Grundgerüst von 12 Spalten.
- Eine verschachtelbare Elementestruktur beliebig vieler Ebenen.
- Geräteabhängige Responsivität, vielleicht ein fluides Design, anstelle eines adaptiven mit nur einigen Größen.
- Veränderbarkeit. Komplizierte und fehleranfällige Umrechnungen von Grundmaßen von 4.4% nach 3.6% sollten vermieden werden.
- Abwärtskompatibilität (bis zu einem gewissen Punkt). Zum Beispiel Android, iOS 4+, Webkit Browsers, Firefox und IE8+.

# ENTWURF EINER RESPONSIVE-GRID.CSS

```
.row { }  
.column, .columns { }  
.row .one { }  
.row .two { }  
.row .three { }  
.row .four { }  
.row .five { }  
.row .six { }  
.row .seven { }  
.row .eight { }  
.row .nine { }  
.row .ten { }  
.row .eleven { }  
.row .twelve { }
```

# DIE HTML STRUKTUR

```
<div class="row">
  <div class="four columns"> ... </div>
  <div class="four columns"> ... </div>
  <div class="four columns"> ... </div>
</div>
```

# DIE BREITE DES LAYOUTS DEFINIEREN

```
.row {  
    width: 960px;      /* Setze die Spaltenbreite auf 960px */  
    max-width: 100%;   /* wenn das Fenster groß genug ist, */  
    min-width: 768px;  /* Fenster darf nicht zu klein werden. */  
    margin: 0 auto;    /* Altmodisch für "Zentrieren" */  
}  
.column, .columns {  
    float: left;       /* Jede Spalte muss floaten! */  
    min-height: 1px;    /* und benötigt eine Mindesthöhe, um die  
                        Reihenfolge des floatens korrekt  
                        umzusetzen */  
    padding: 0 15px;    /* Abstand zwischen Rahmen und Inhalt */  
    position: relative;  
}
```

# DIE EINZELNEN SPALTENBREITEN

Das Prozentmaß basiert auf dem Grundmaß der Klasse `.row!`

```
.row .one    { width: 8.3333333%; }
.row .two    { width: 16.666667%; }
.row .three   { width: 25%; }
.row .four    { width: 33.333333%; }
.row .five    { width: 41.666667%; }
.row .six    { width: 50%; }
.row .seven   { width: 58.333333%; }
.row .eight   { width: 66.666667%; }
.row .nine    { width: 75%; }
.row .ten    { width: 83.333333%; }
.row .eleven   { width: 91.666667%; }
.row .twelve   { width: 100%; }
```

# MIT BOX-SIZING DAS BOX MODEL UMSTELLEN

```
body * {  
  box-sizing: border-box;  
}
```

Gewöhnlich wird der Padding Wert zur Breite eines Blockelements hinzugerechnet, ebenso die Borderbreite. Das ist aber nicht sehr praktisch.

Das IE 6 Standard Boxmodel „border-box“ schließt weder Padding noch Border in die Elementenbreite mit ein.

# BOX-SIZING FÜR ALLE BROWSER

```
* {  
  -moz-box-sizing: border-box;  
  -webkit-box-sizing: border-box;  
  -o-box-sizing: ...  
  -khtml-...  
  -ms-...  
  box-sizing: border-box;  
}
```

Mehr über die border-box unter <http://paulirish.com/2012/box-sizing-border-box-ftw/>

# VERSCHACHTELBARKEIT DER ELEMENTE

Nach der `.row` Declaration einfügen:

```
.row .row {  
    width: auto;      /* Die Spalte passt sich der  
                      Containerspalte an */  
    max-width: none;  
    min-width: 0;  
    margin: 0 -15px; /* Ein negatives left/right Margin  
                      kompensiert das Padding des  
                      Containerelementes */  
}
```

# HTML KANN NUN FOLgendes ...

```
<div class="row">
  <div class="four columns"> // width: 320px;
    <div class="row"> // width: auto;
      <div class="three columns"> ... </div> //80px
      <div class="nine columns"> ... </div>
    </div>
  </div>
<div class="four columns"> ... </div>
<div class="four columns"> ... </div>
</div>
```

# DAS MICROCLEARFIX VON NICOLAS GALLAGHER

```
.row:before, .row:after,  
.clearfix:before, .clearfix:after  
{  
    content : " ";  
    display : table;  
}  
.row:after, .clearfix:after { clear: both; }  
.row, .clearfix { *zoom: 1; }
```

# EINE LÖSUNG FÜR DAS SUBPIXEL-PROBLEM

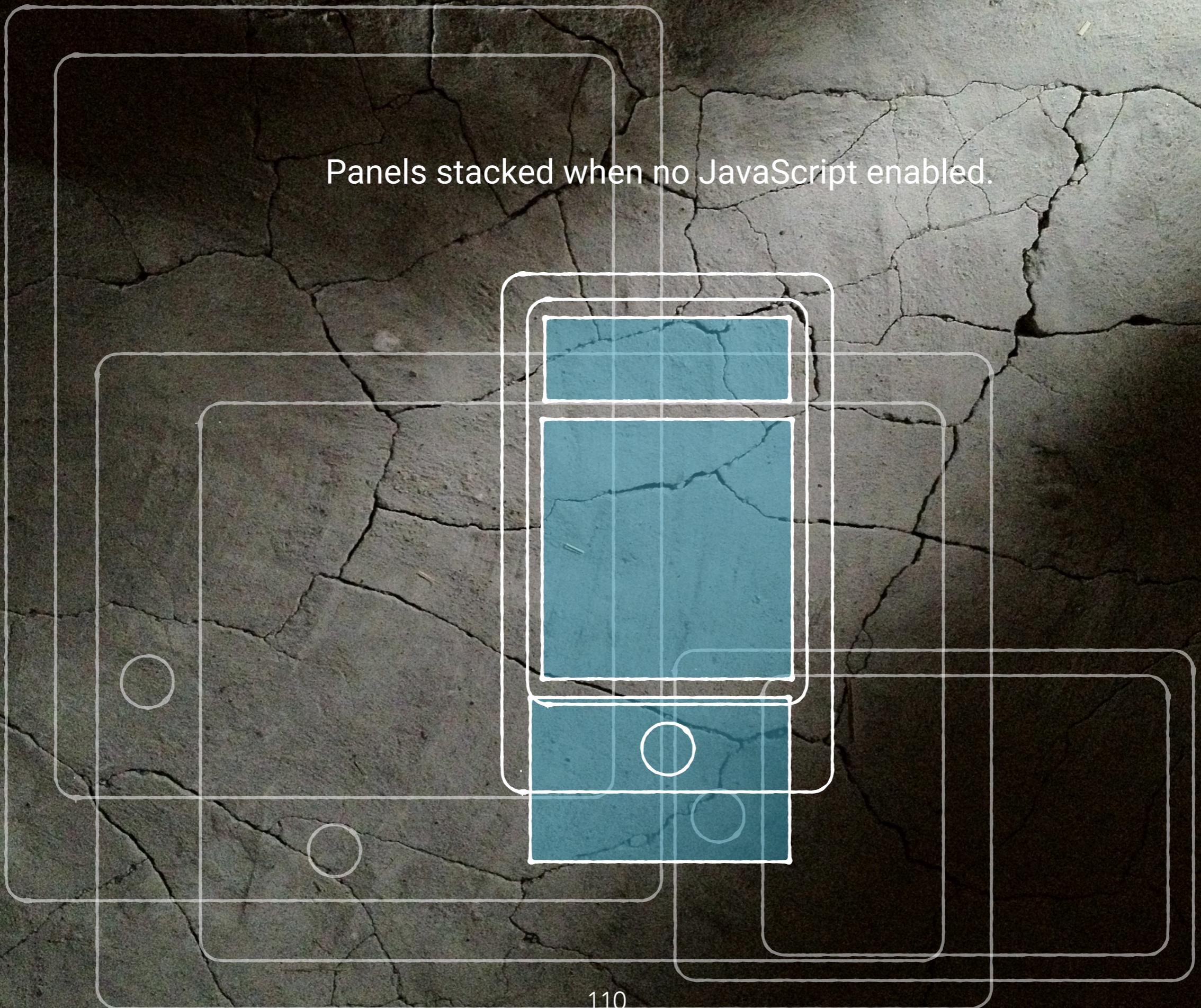
```
[class*="column"] + [class*="column"]:last-child {  
    float: right;  
}  
  
[class*="column"] + [class*="column"].end {  
    float: left;  
}
```

<http://www.netmagazine.com/tutorials/building-modern-grid-system>.

# EIN OFF CANVAS LAYOUT

```
[role="navigation"],  
[role="main"],  
[role="complementary"] {  
    transition: .2s all ease;  
    width: 90%;  
    padding: 5%;  
}
```

Panels stacked when no JavaScript enabled.

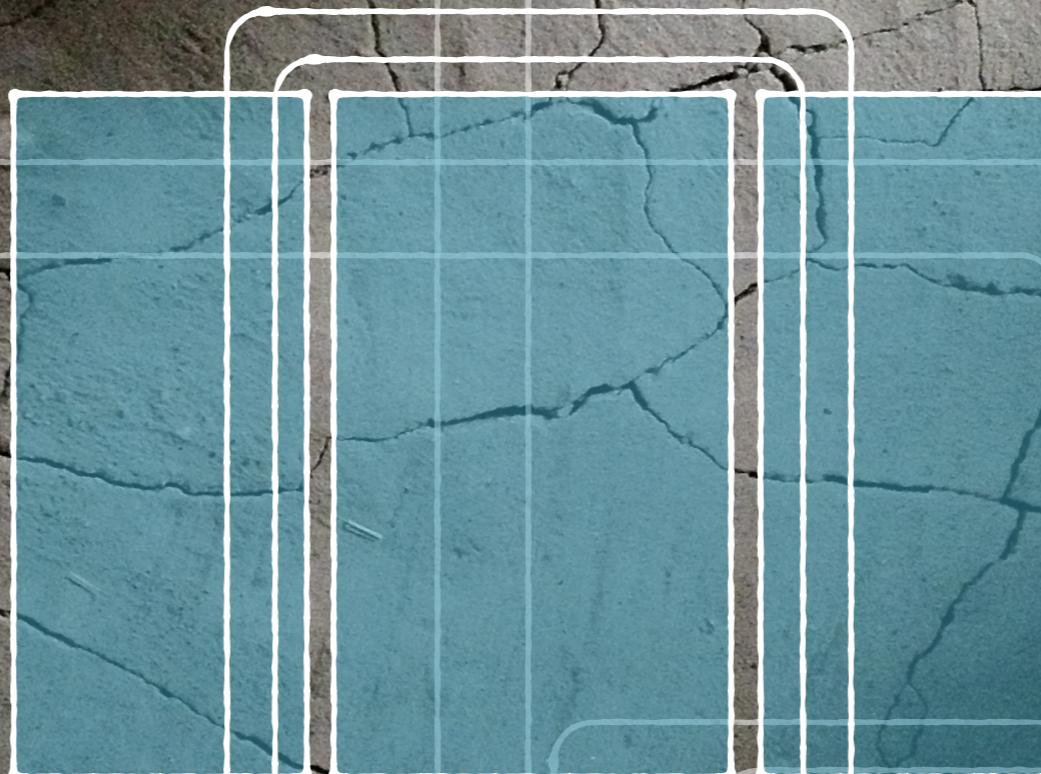


# A RESPONSIBLE START

If JavaScript is detected, declare our layout styles for each panel with the navigation to the left and the sidebar content to the right.

```
.js [role="navigation"] {  
  margin-left: -100%;  
  float: left;  
}  
.js [role="main"] {  
  margin-left: 0;  
  float: left;  
}  
.js [role="complementary"] {  
  margin-right: -200%;  
  float: left;  
}
```

Mobile-First styles centered by default.

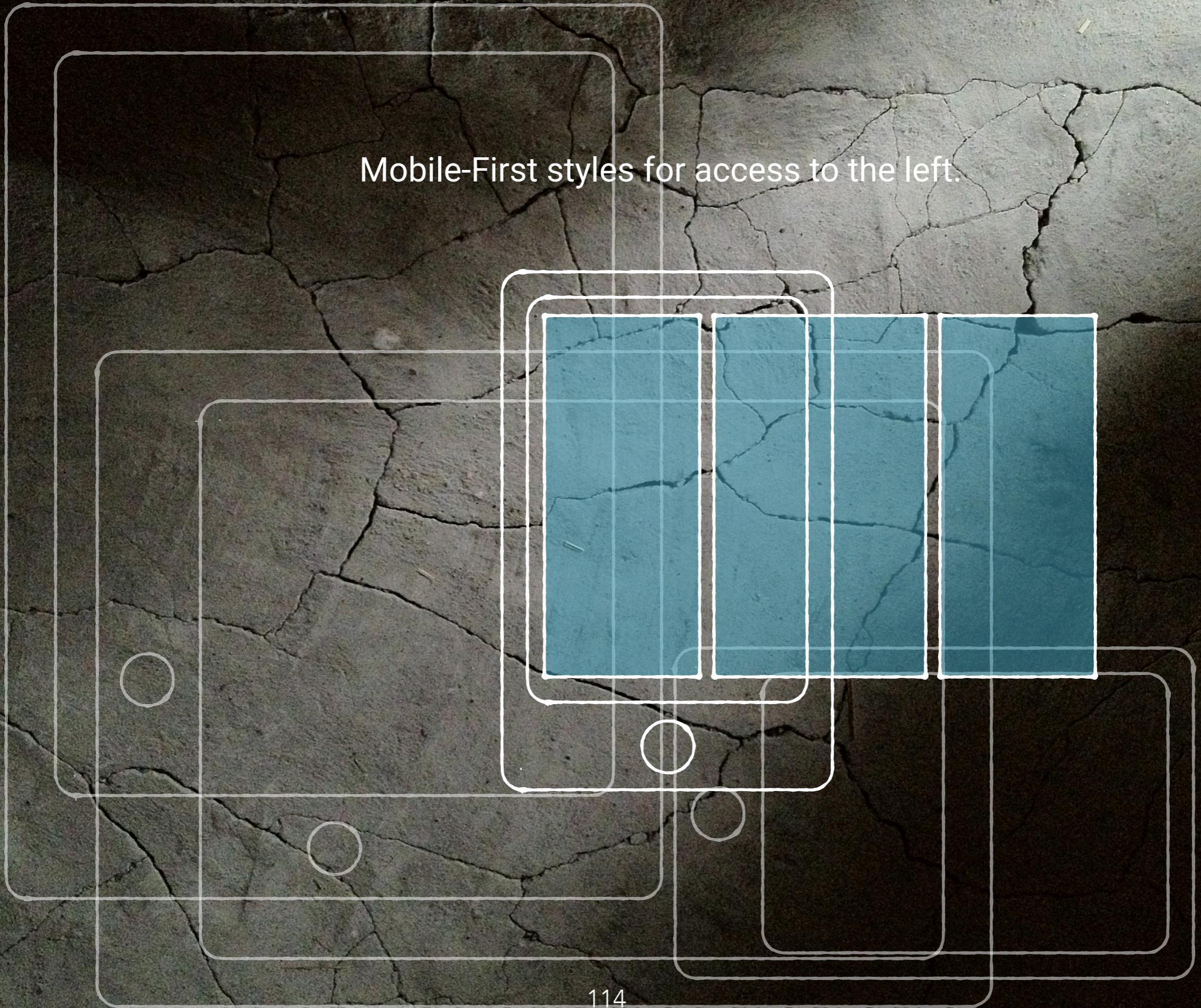


# ACCESS LEFT

For this demo, I'm using the left panel as navigation. We'll need an anchor element that will default to jump down the page to the `#nav` id. To get this to open the panel, we'll use a bit of jQuery to prevent default plus add a class to the body when clicked. Let's take a look at the panel styles for when the navigation is open.

```
.active-nav [role="navigation"] {  
    margin-left: 0;  
    width: 80%;  
}  
.active-nav [role="main"]{  
    margin-right: -100%;  
}  
.active-nav [role="complementary"] {  
    margin-right: -100%;  
    float: right;  
}
```

Mobile-First styles for access to the left.

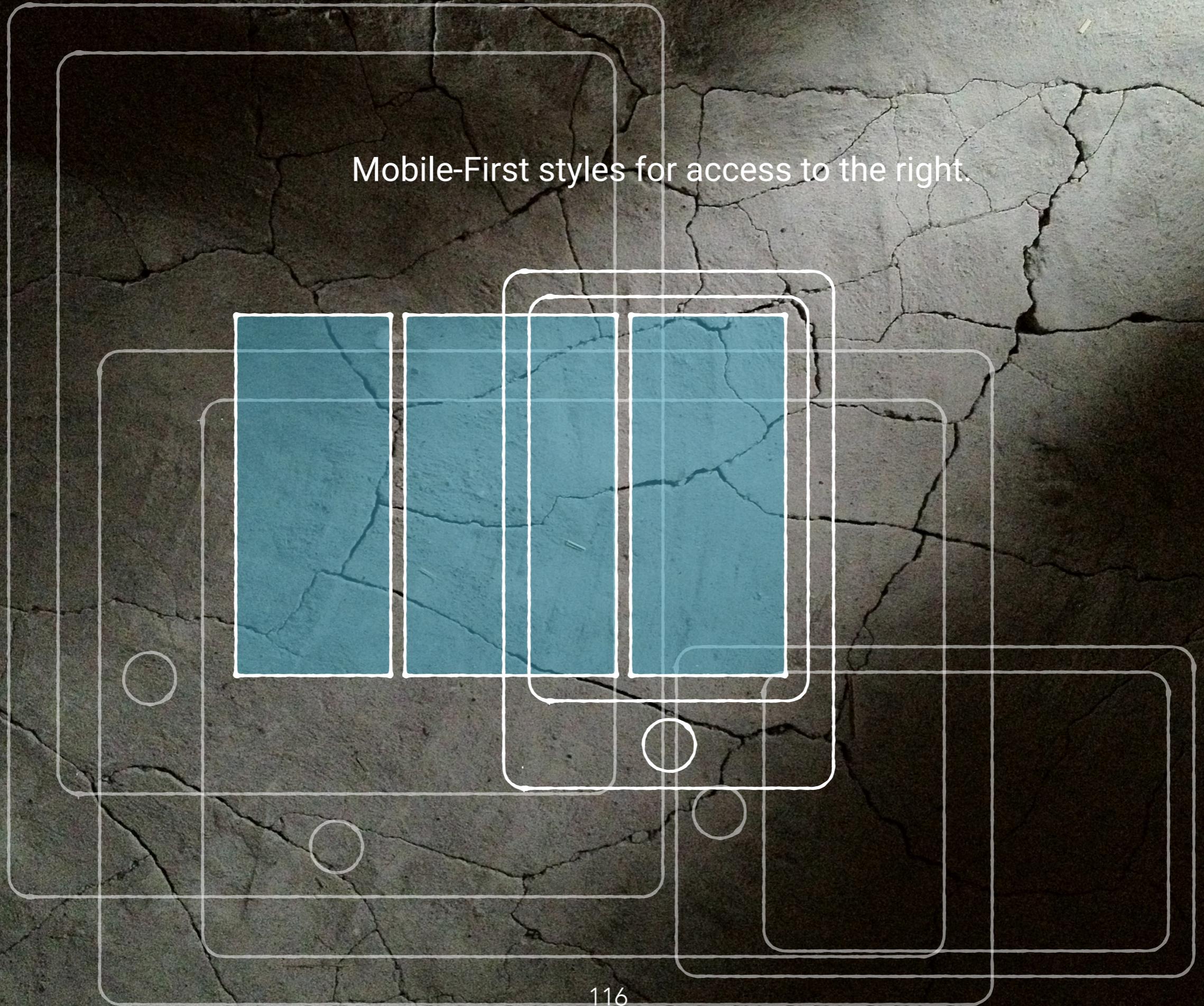


# ACCESS RIGHT

Now we take the same concept to the other side for the hidden sidebar panel. We'll use slightly different styles.

```
.active-sidebar [role="navigation"] {  
    margin-left: -100%;  
}  
.active-sidebar [role="main"] {  
    margin-left: -90%;  
}  
.active-sidebar [role="complementary"] {  
    margin-left: 0;  
    width: 80%;  
}
```

Mobile-First styles for access to the right.

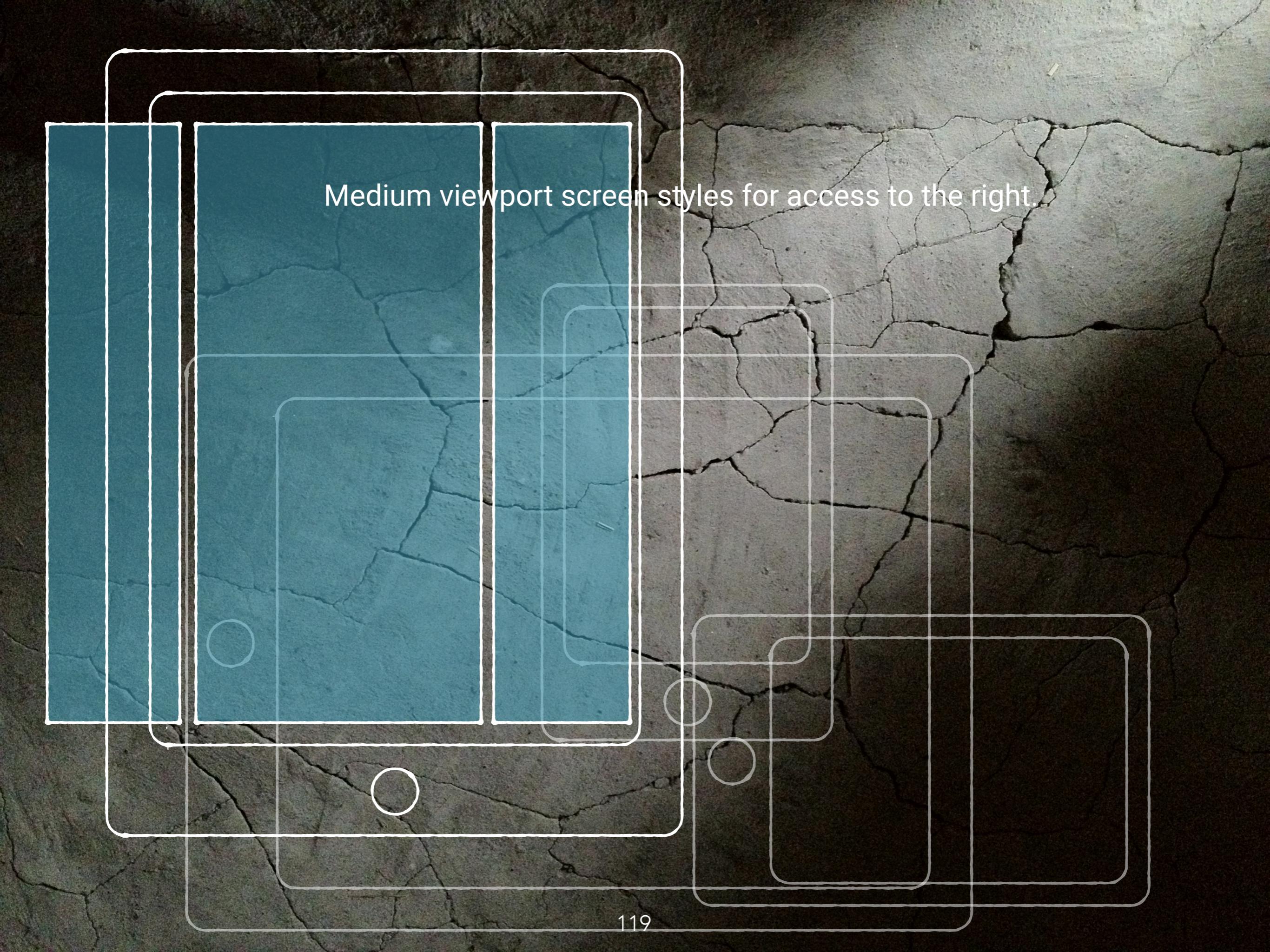


# MEDIUM VIEWPORT STYLES

For medium size viewports, we'll have the navigation panel show by default and hide the sidebar.

```
@media all and (min-width: 600px) {  
  .js [role="navigation"] {  
    width: 20%;  
    margin-left: 0;  
  }  
  .js [role="main"] {  
    width: 60%;  
    float: left;  
  }  
  .js [role="complementary"] {  
    width: 20%;  
  }  
  .active-sidebar [role="navigation"] {  
    margin-left: -100%;  
  }  
  .active-sidebar [role="main"] {  
    margin-left: 0;  
    width: 60%;  
  }  
  .active-sidebar [role="complementary"] {  
    margin-right: -80%;  
    width: 20%;  
  }  
}
```

Medium viewport screen styles showing left and main panel by default.



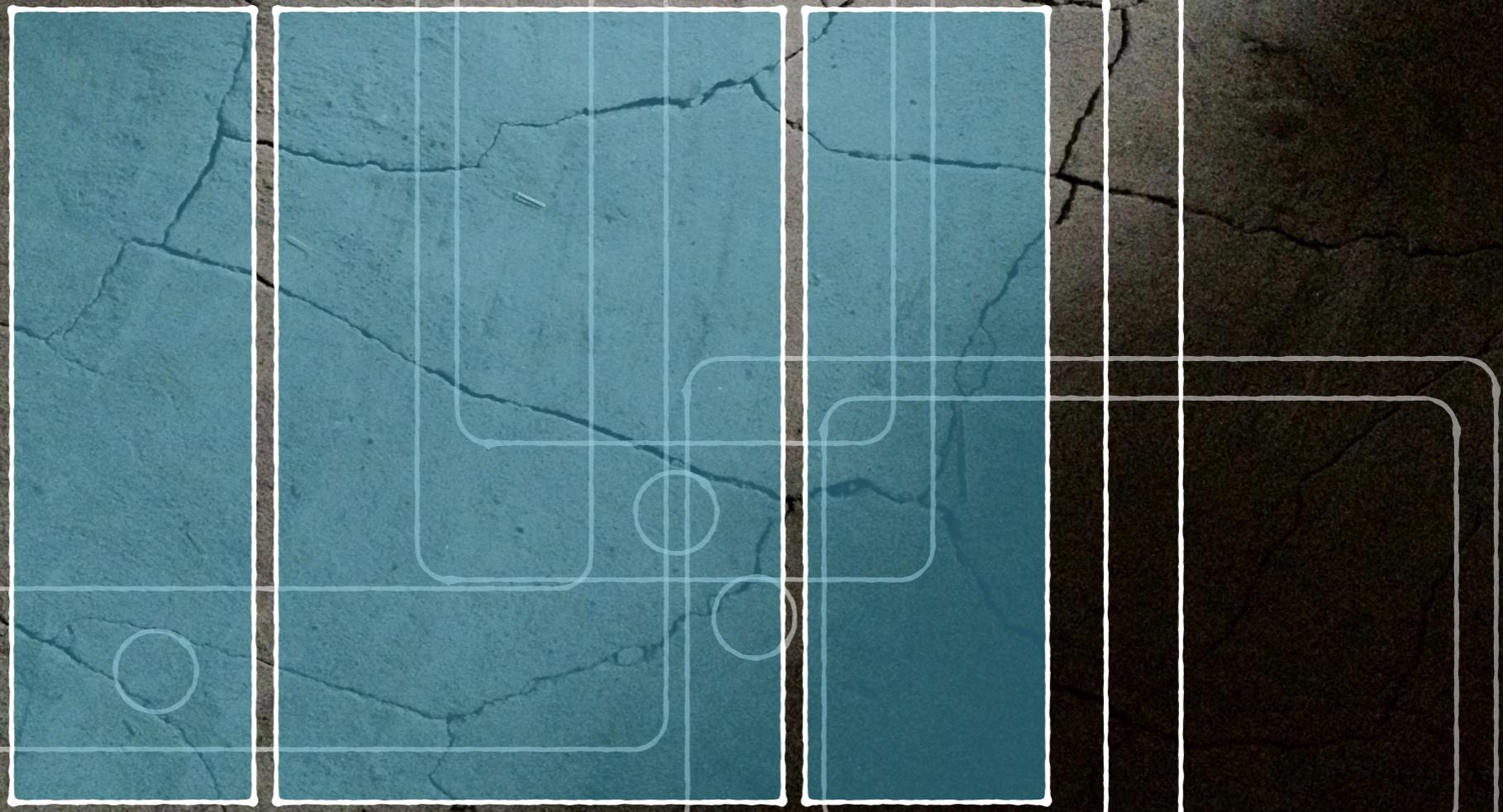
Medium viewport screen styles for access to the right.

# LARGE VIEWPORT STYLES

For large size viewports, we'll show all panels.

```
@media all and (min-width: 800px) {  
  .js [role="navigation"] {  
    width: 19%;  
    margin-left: 0;  
    float: left;  
    padding: 3%;  
  }  
  
  .js [role="main"] {  
    width: 44%;  
    padding: 3%;  
  }  
  
  .js [role="complementary"] {  
    width: 19%;  
    padding: 3%;  
    margin-right: 0;  
    float: right;  
  }  
}
```

Large viewport screen styles showing all panels by default.



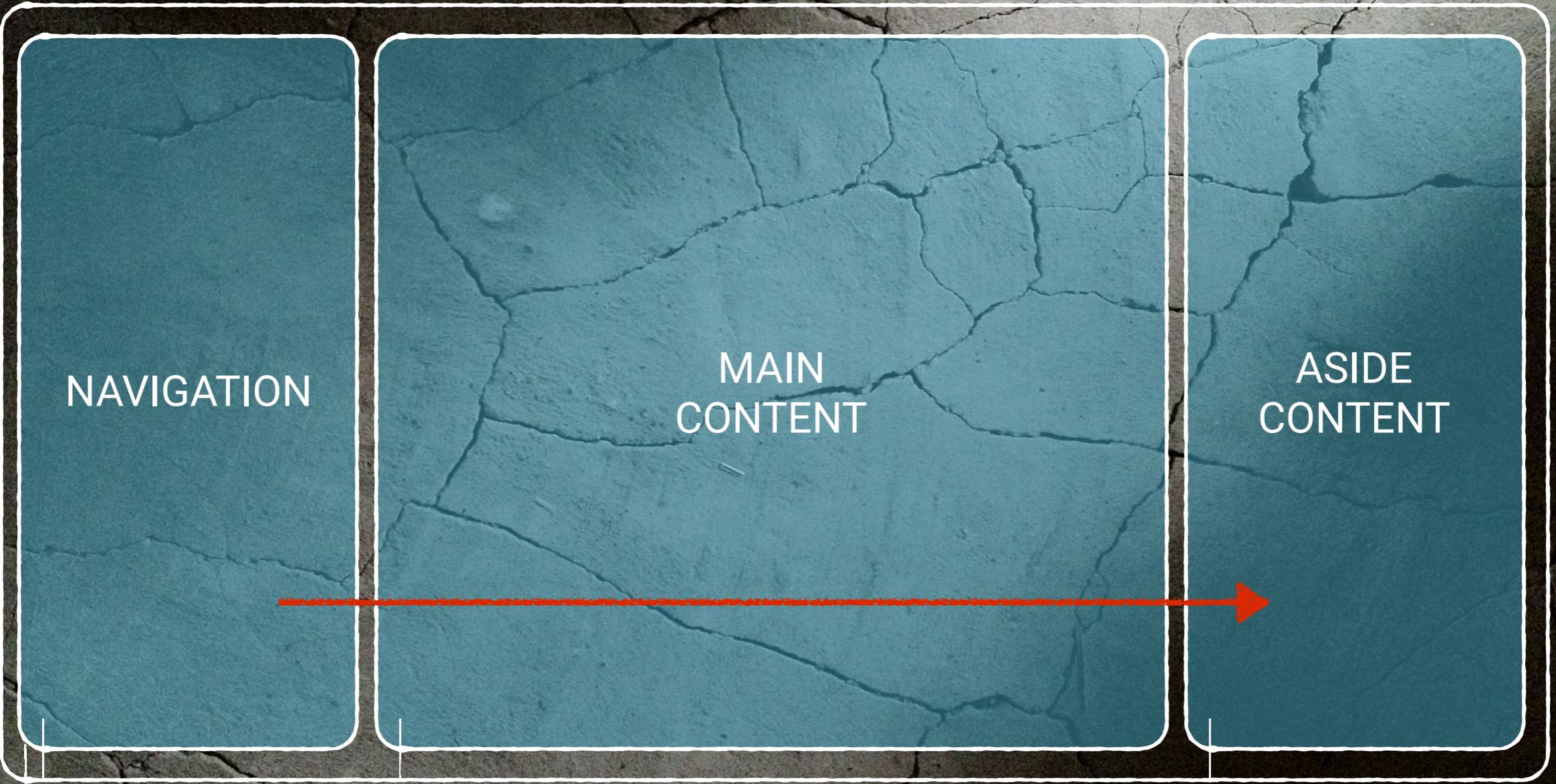
<http://jasonweaver.name/lab/offcanvas>

– JASON WEAVER

# NEUES LAYOUTEN FLEX BOX

# WAS SIND FLEX BOXES?

- Flex boxes bieten einen sehr effizienten Weg, um Layoutelemente anzuordnen.
- In einem „Container“ werden „Items“ angeordnet, ausgerichtet und verteilt, auch wenn deren Größe nicht bekannt oder dynamisch ist.
- Responsive Layouts lassen sich mit flex boxes einfacher und stabiler umsetzen, als mit dem bisherigen float/clear Verfahren.



# FLEX CONTAINER

```
<div class="page" id="news">  
  <div class="page-header"> . . . </div>  
  <div class="page-content">  
    <div class="content-nav"> . . . </div>  
    <div class="content-main"> . . . </div>  
    <div class="content-aside"> . . . </div>  
  </div>  
  <div class="page-footer"> . . . </div>  
</div>
```

# FLEX ITEMS

```
<div class="page" id="news">  
  <div class="page-header"> . . . </div>  
  
  <div class="page-content">  
    <div class="content-nav"> . . . </div>  
    <div class="content-main"> . . . </div>  
    <div class="content-aside"> . . . </div>  
  </div>  
  
  <div class="page-footer"> . . . </div>  
</div>
```

# DER FLEXBOX-CONTAINER

```
.page-content {  
    display: flex; /* Kindelemente werden Items */  
  
    flex-direction: row; /* Items zeilenweise anordnen */  
    flex-wrap: nowrap; /* Items dürfen nicht umbrechen */  
    flex-flow: /* Shorthand für flex-direction/flex-wrap */  
  
    align-items: stretch; /* Höhe der Items anpassen */  
    align-content: flex-start; /* vertikale Ausrichtung der Items */  
  
    justify-content: space-between; /* horizontale Itemausrichtung */  
}
```

# FLEX-ITEMS

```
.content-nav {  
    flex-basis : 25%; /* initiale Breite des Elements */  
    flex-grow : 1;    /* Proportion beim Vergrößern */  
    flex-shrink : 1;  /* Proportion beim Verkleinern */  
}  
.content-main {  
    flex-basis : 50%;  
    flex-grow : 2;  
    flex-shrink : 2;  
}  
.content-aside {  
    flex-basis : 25%;  
    flex-grow : 1;  
    flex-shrink : 1;  
}
```

Andere Eigenschaften für Items sind: `order` (Reihenfolge) und `align-self` (Vertikale Ausrichtung des Items)

<http://css-tricks.com/snippets/css/a-guide-to-flexbox/>

- CHRIS COYIER

# A FLUID „HOLY-GRAIL“ LAYOUT

```
/**  
 * LAYOUT STYLES  
 */  
  
/* BASE  
   the root container must have a 100% height */  
root,  
html, body {  
  display : flex;  
  flex-direction: column;  
  height : 100%;  
  min-height : 100%;  
}  
  
/* LAYOUT */  
.page { display : flex; width : 100%; flex-direction : column; height : 100%; min-height : 100%; }  
  
.page-header { display : block; width : 100%; flex-grow : 0; flex-shrink : 0; }  
.page-content { display : flex; width : 100%; flex-direction : row; flex-grow : 1; flex-shrink : 0; }  
.page-footer { display : block; width : 100%; flex-grow : 0; flex-shrink : 0; }  
  
.content-nav { display : block; flex-basis : 25%; flex-grow : 1; flex-shrink : 1; }  
.content-main { display : block; flex-basis : 50%; flex-grow : 2; flex-shrink : 2; }  
.content-aside { display : block; flex-basis : 25%; flex-grow : 1; flex-shrink : 1; }  
  
/* MARGINS AND PADDINGS */  
.page { display : flex; width : 100%; flex-direction : column; height : 100%; min-height : 100%; }  
  
.page-header { padding : 0.5rem; }  
.page-content { padding : 0; }  
.page-footer { padding : 0.5rem; }  
.content-nav { padding : 0.5rem; }  
.content-main { padding : 0.5rem; }  
.content-aside { padding : 0.5rem; }  
  
.content-nav > * { margin : -0.5rem; }
```

# RASTER PER CSS DEKLARATION CSS GRIDS

# WAS SIND GRIDS?

- Mit Grids kann Seitenraster direkt per CSS festgelegt werden.
- Es wird zunächst eine horizontale/vertikale Einteilung eines Layoutelements festgelegt. Dann werden die darin befindlichen Elemente zwischen Grid-Linien "aufgespannt".
- Ausserdem können Satzbereiche (Flächen) definiert und verwendet werden.

# Can I use

# grid



Settings

1 result found

## CSS Grid Layout - CR

Global

0.04% + 5.86% = 5.9%

unprefixed:

0.04%

Method of using a grid concept to lay out content, providing a mechanism for authors to divide available space for lay out into columns and rows using a set of predictable sizing behaviors

1

Current aligned

Usage relative

Date relative

Show all

IE

Edge

\* Firefox

Chrome

Safari

Opera

iOS Safari

\* Opera Mini

\* Android Browser

Chrome for Android

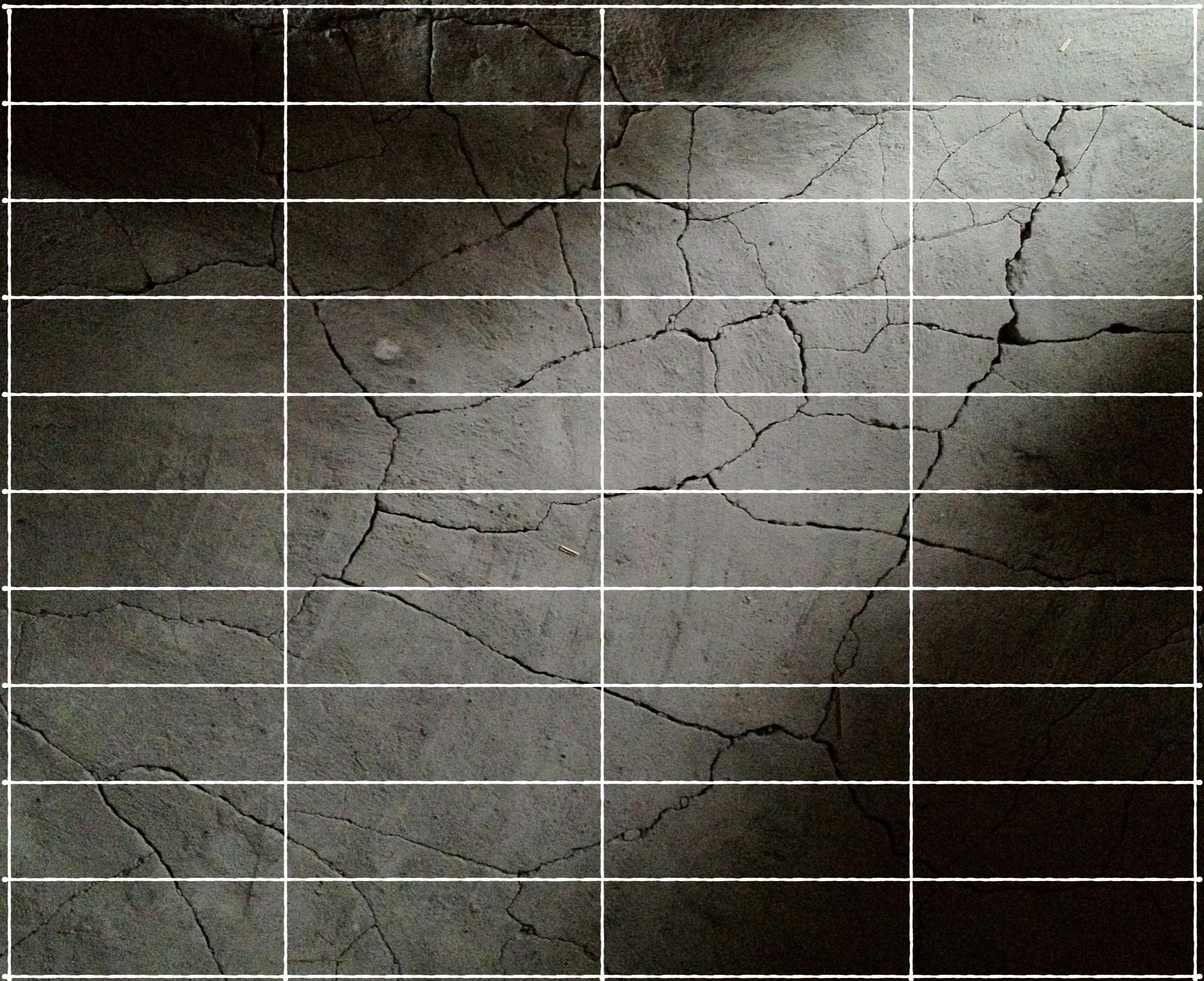
				49					4.3		
				51					4.4		
				54				9.3	4.4.4		
2	11		2	14		3	50	10	42	10.2	all
			2	15		3	51	TP	43		53
						52	57		44		55
				53			58				

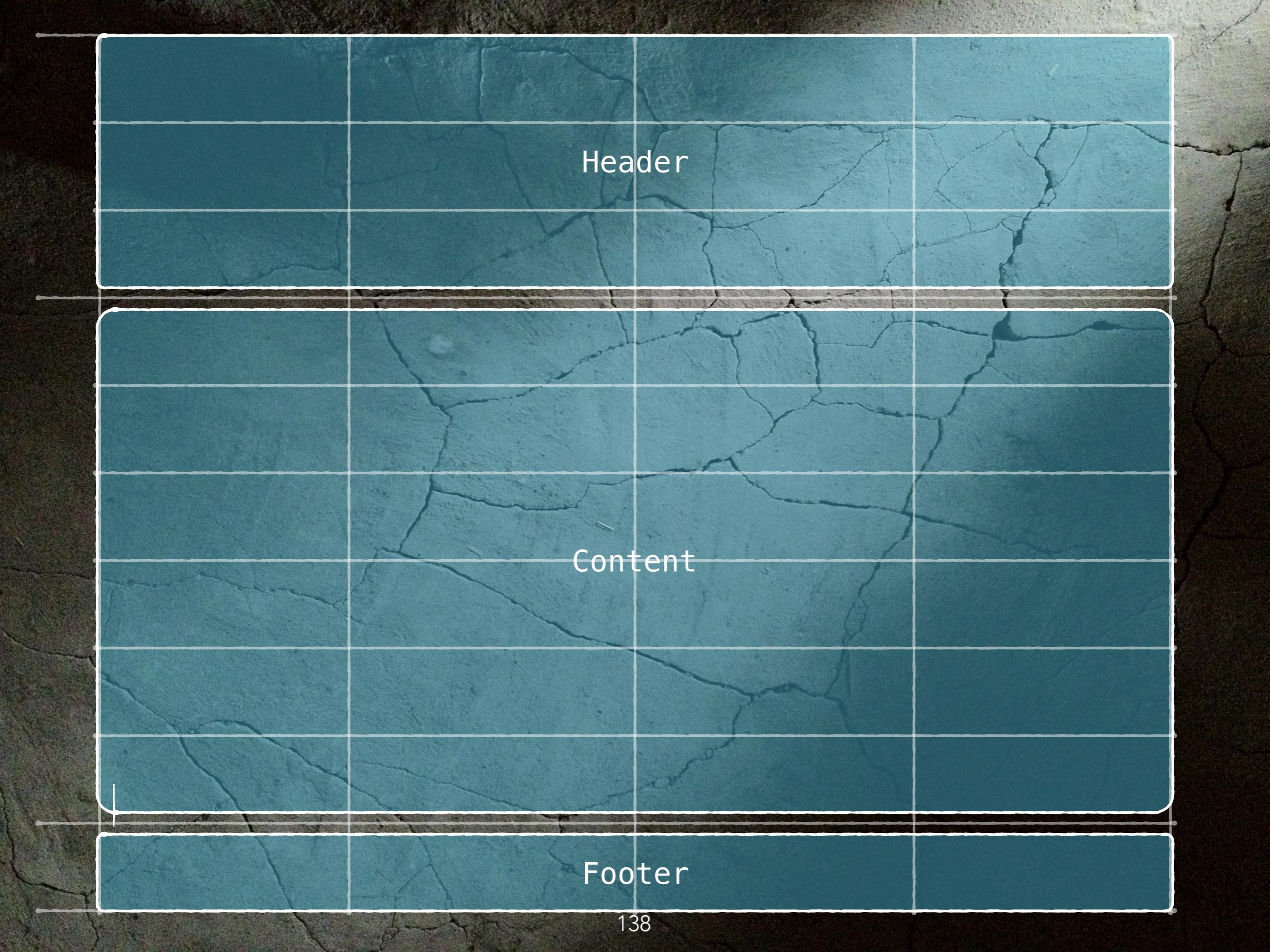
# FLEX CONTAINER

```
<section>
  <div class="page">
    <div class="page-header">
      <header></header>
    </div>
    <div class="page-content">
      <div class="content-nav"> . . . </div>
      <div class="content-main"> . . . </div>
      <div class="content-aside"> . . . </div>
    </div>
    <div class="page-footer"> . . . </footer>
  </div>
</section>
```

# FLEX ITEMS

```
▪ page {  
    display: grid;  
    grid-template-columns: 25vw 25vw 25vw 25vw;  
    grid-template-rows: 10vh 10vh 10vh 10vh 10vh 10vh 10vh 10vh 10vh 10vh;  
}
```





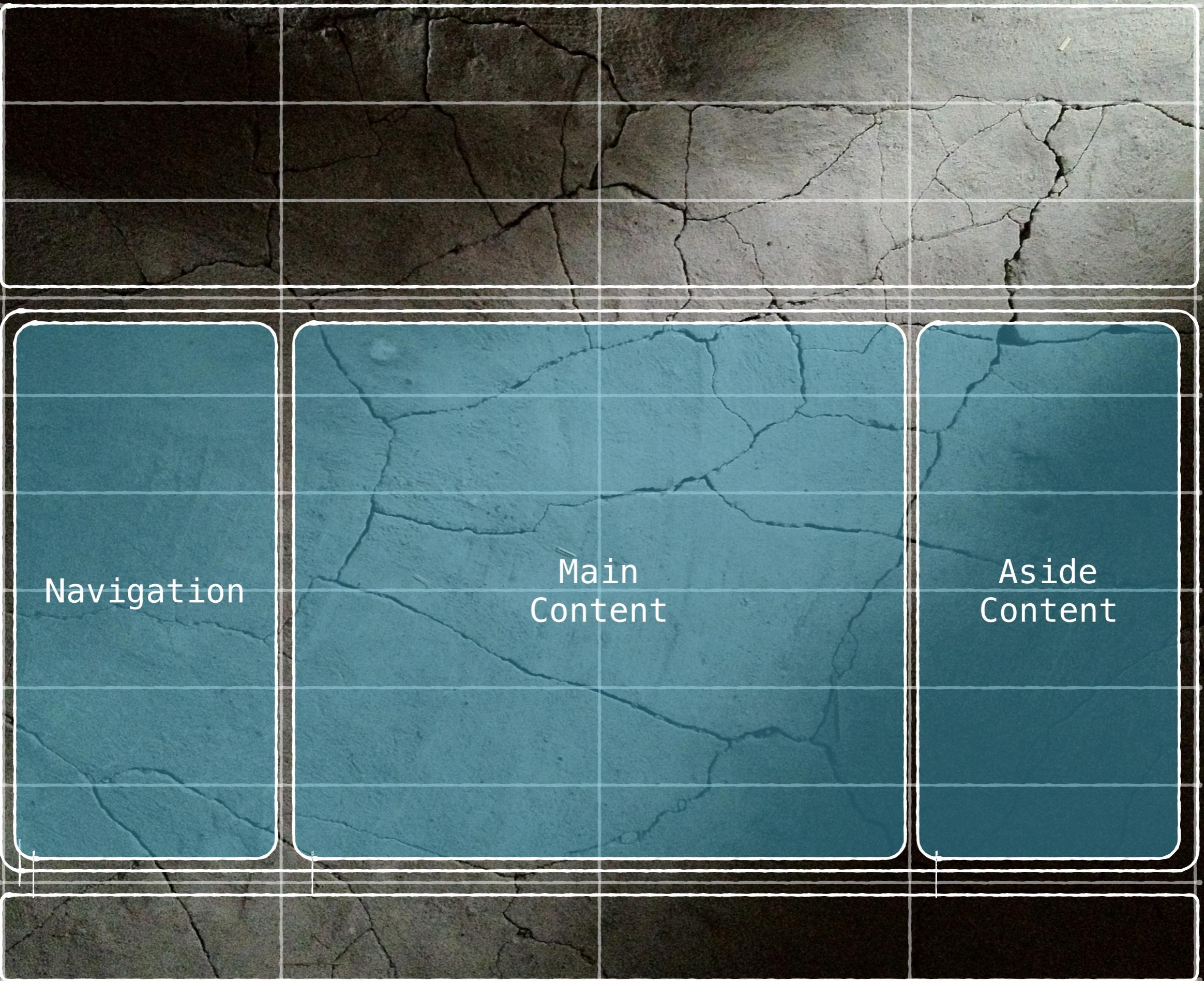
Header

Content

Footer

# FLEX ITEMS

```
.page-content, .page-footer, .page-header {  
    grid-column-start: 1;  
    grid-column-end: 5;  
}  
  
.page-header {  
    grid-row-start: 1;  
    grid-row-end: 3;  
}  
  
.page-content {  
    grid-row-start: 3;  
    grid-row-end: 10;  
}  
  
.page-footer {  
    grid-row-start: 10;  
    grid-row-end: 11;  
}
```



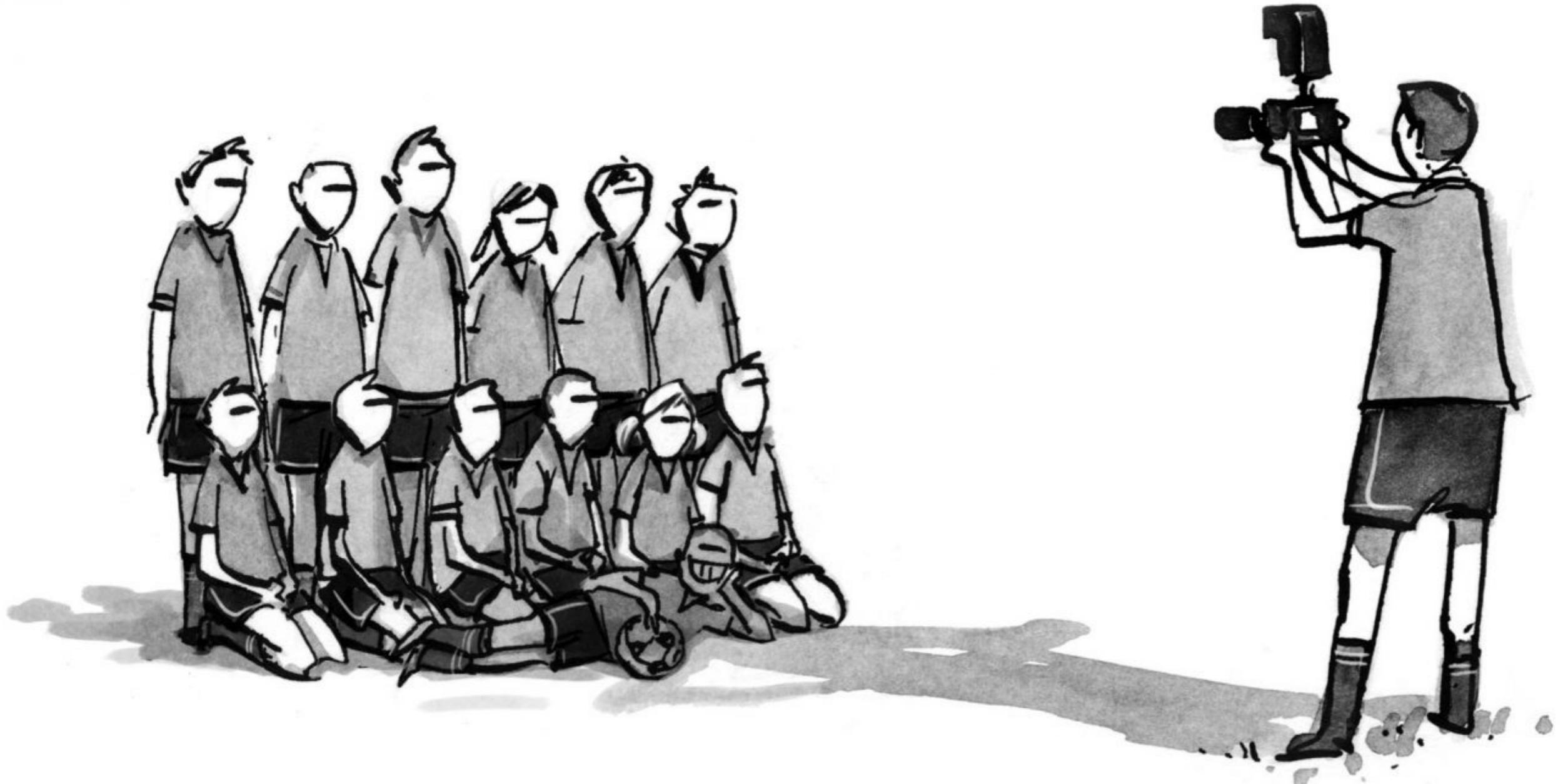
# DIE GRID DEFINITION UND ANWENDUNG

```
.page-content {  
    display: grid;  
    grid-template-columns: 25vw 25vw 25vw 25vw;  
    grid-template-rows: 10% 10% 10% 10% 10% 10% 10% 10% 10%;  
}  
  
.content-aside, .content-main, .content-nav {  
    grid-row-start: 1;  
    grid-row-end: 11;  
}  
  
.content-nav {  
    grid-column-start: 1;  
    grid-column-end: 2;  
}  
  
.content-main {  
    grid-column-start: 2;  
    grid-column-end: 4;  
}  
  
.content-aside {  
    grid-column-start: 4;  
    grid-column-end: 5;  
}
```

<https://css-tricks.com/snippets/css/complete-guide-grid/>

- CHRIS HOUSE

# GERÄTE UNTERSCHIEDEN RESPONSIVE WEBDESIGN



<http://alistapart.com/article/responsive-web-design>

-ETHAN MARCOTTE

# INHALTE FÜR GERÄTE UNTERSCHIEDEN RESPONSIVE HAT VIELE EBENEN!

Responsive  
Templates

Responsive  
Typography

Responsive  
Images

Responsive  
Content

Responsive  
Behavior

...

# GERÄTE UNTERSCHIEDEN RESPONSIVE TEMPLATES

# MEDIEN ATTRIBUTE EINSETZEN

```
@media screen and (max-width:480px) { ... }
@media screen and (min-width:481px) { ... }
@media screen and (min-width:768px) { ... }
@media screen and (min-width:1024px) { ... }
@media screen and (min-width:1280px) { ... }
@media screen and (min-width:1600px) { ... }
@media screen and (min-width:1920px) { ... }
```

# MOBILE FIRST PROGRESSIVE ENHANCEMENT

## ALL DEVICES

```
@media screen {  
    declarations for all devices,  
    optimized for xtra small mobile devices  
}
```

## XTRA SMALL

```
@media screen and (max-width:767px) {  
    changes/specials for extra small  
}
```

## SMALL

```
@media screen and (min-width:768px) { ... }
```

## MEDIUM

```
@media screen and (min-width:992px) { ... }
```

## LARGE

```
@media screen and (min-width:1200px) { ... }
```

# DESKTOP FIRST GRACEFUL DEGRADATION

## ALL DEVICES

```
@media screen {  
    declarations for all devices,  
    optimized for medium devices  
}
```

## MEDIUM

```
@media screen and (min-width:1024px) and (max-width:1279px) {  
    changes/specials for medium  
}
```

SMALL	@media screen and (max-width:767px) { ... }
XTRA SMALL	@media screen and (max-width:480px) { ... }
LARGE	@media screen and (min-width:1280px) { ... }
XTRA LARGE	@media screen and (min-width:1600px) { ... }
HD	@media screen and (min-width:1920px) { ... }

# ANDERE ATTRIBUTE

```
@media screen and (orientation:portrait) { ... }  
  
@media print and (resolution:300dpi) { ... }  
  
@media screen and (min-resolution:96dpi) { ... }  
@media screen and (min-resolution:192dpi) { ... } // Retina  
  
@media print and (color) { ... }  
  
@media screen and (aspect-ratio:16/9) { ... }
```

# DIE WICHTIGEN ÄNDERUNGEN UND ERNEUERUNGEN GERÄTEERKENNUNG MIT CSS UND MEDIAQUERIES

# MEDIA QUERIES MIT ATTRIBUTEN

---

- HTML5 präzisiert Mediatypen.
- Erweiterung durch Media Attributes wie 'width', und 'orientation'
- Damit können medienabhängige Stylesheets angesprochen werden.

# MEDIENTYPEN WERDEN BEREITS IN HTML 4 DEFINIERT

• screen	Bildschirme	
projection	Beamer	wird nicht benutzt
handheld	kleine Computer	wird nicht benutzt
tv	Fernsehgeräte	wird nicht benutzt
• print	Drucker	
tty	Nadeldrucker	
• aural	auditiv	deprecated seit CSS2
braille	Blindenschrift	
embossed	Blindenschrift	seit CSS2
speech	Sprachausgabe	seit CSS2
• all	Alle Medientypen	seit CSS2

# EXTERNES MEDIENTYP- ABHÄNGIGES STYLESHEET

```
<link  
    rel="stylesheet"  
    type="text/css"  
    media="screen"  
    href="sans-serif.css"
```

v

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

v

# MEDIABLOCK IN EINER CSS DATEI

```
@media screen {  
    html { font-family: sans-serif }  
    ...  
}
```

# IN HTML5 GIBT ES ABFRAGBARE MEDIENEIGENSCHAFTEN

- width, height,
- device-width, device-height
- orientation
- aspect-ratio, device-aspect-ratio
- color, color-index, monochrome
- resolution
- scan
- grid

# EXTERN ODER INTERNE MEDIENTYPISIERUNG

```
<link  
    rel="stylesheet"  
    type="text/css"  
    media="screen and (color)"  
    href="example.css"  
    >  
  
@media screen and (color) { ... };
```

# 'ALL' UMFASTT ALLE MEDIENTYPEN

---

'all' ist meist implizit  
und umfasst alle Media Typen

`@media (orientation: portrait) { ... }`

ist das gleiche, wie:

`@media all and (orientation: portrait)  
{ ... }`

# NOT ALS NEGATION

```
<link  
    rel="stylesheet"  
    type="text/css"  
    media="not screen and (color)"  
    href="example.css"
```

v

# ONLY SCREEN

Das Keyword 'only' kann verwendet werden, um ein Stylesheet vor älteren User Agenten zu verbergen.  
Neue Browser ignorieren das Wort einfach.

```
<link  
  rel="stylesheet"  
  media="only screen and (color)"  
  href="example.css"
```

v

# MEHRERE MEDIA QUERIES ALS KOMMASEPARIERTE LISTE

```
@media  
screen and (color),  
projection and (color)  
{  
...  
}
```

# MEDIAQUERIES MIT FEHLERN

Wenn ein Media Feature auf ein Ausgabegerät nicht zutrifft, so liefert der User Agent ein false. Das Query wird abgelehnt.

```
<link  
  rel="stylesheet"  
  media="aural and (device-aspect-ratio: 16/9)"  
  href="example.css" v>
```

```
<link  
  rel="stylesheet"  
  media="speech and (min-device-width: 800px)"  
  href="example.css" v>
```

Einsatz und Anwendungsbeispiele.

# MEDIENATTRIBUTE.

# WIDTH, HEIGHT

- Wert: <length>, <height>
- Anwendbar für visuelle und taktile Medientypen.
- Kann mit min/max kombiniert werden.
- 'width', 'height' bezeichnen die Breite/Höhe des Darstellungsbereiches.
- Für kontinuierliche Medien sind das Breite und Höhe des Viewports inklusive der Scrollbar.
- Für Medien, die mit Seiten arbeiten, sind das Breite und Höhe des Seitenlayouts oder der Textbox.
- Breiten und Höhen dürfen nicht negativ sein.

# FÜR DRUCKSEITEN MIT EINER BREITE GRÖßER 25 CM.

- <link  
  rel="stylesheet"  
  media="print and (min-width: 25cm)"  
  href="http://...">
- Für Geräte mit einem Viewport zwischen 400 und 700 Pixeln:
  - @media  
  screen  
  and (min-width: 400px)  
  and (max-width: 700px)  
  { ... }

# DEVICE-WIDTH, DEVICE-HEIGHT

- Wert: <length>, <height>
- Anwendbar für visuelle und taktile Medientypen.
- Kann mit min/max kombiniert werden.
- 'device-width' und 'device-height' bezeichnen die Breite/Höhe des Ausgabefläche. Bei kontinuierlichen Medien ist das die Bildschirmbreite. Bei seitenbasierten Medien ist das die Blattgröße.
- Breiten und Höhen dürfen nicht negativ sein.

# ORIENTATION

- Value: portrait | landscape
- Anwendbar für bitmap basierte Medien
- Kann nicht mit min/max kombiniert werden.
- 'orientation' ist 'portrait', wenn die Höhe größer oder gleich ist, als die Breite.
- Sonst gilt 'landscape'.

# ANWENDUNG DER ORIENTATION

- @media all and (orientation:portrait) { ... }
- @media all and (orientation:landscape) { ... }

# ASPECT-RATIO

- Value: <ratio>
- Anwendbar für bitmap basierte Medien
- Kann mit min/max kombiniert werden.
- 'aspect-ratio' definiert das Verhältnis von 'width' und 'height' eines Asugabebereiches.

# DEVICE-ASPECT-RATIO

- Value: <ratio>
- Anwendbar für bitmap basierte Medien
- Kann mit min/max kombiniert werden.
- The ‘device-aspect-ratio’ media feature is defined as the ratio of the value of the ‘device-width’ media feature to the value of the ‘device-height’ media feature.

# DAS VERHÄLTNIS 16:9

- @media  
screen and (device-aspect-ratio: 16/9) { ... }
- @media  
screen and (device-aspect-ratio: 32/18) { ... }
- @media  
screen and (device-aspect-ratio: 1280/720) { ... }

# COLOR

- Wert: <integer>
- Anwendbar für visuelle Medientypen.
- Kann mit min/max kombiniert werden.
- 'color' beschreibt die Anzahl von Bits pro Farbkanal des Ausgabegerätes.
- Wenn das Gerät kein Farbgerät ist, dann ist der Wert 0.

# DIE BEIDEN FOLGENDEN QUERIES GELTEN FÜR JEDES FARBAUSGABEGERÄT

- @media all and (color) { ... }
- @media all and (min-color: 8) { ... }

# COLOR-INDEX

- Wert: <integer>
- Anwendbar für visuelle Medientypen.
- Kann mit min/max kombiniert werden.
- The ‘color-index’ media feature describes the number of entries in the color lookup table of the output device. If the device does not use a color lookup table, the value is zero.

# ZWEI WEGE, UM GERÄTE MIT INDIZIERTER FARBAUSGABE ANZUSPRECHEN

- @media all and (color-index) { ... }
- @media all and (min-color-index: 1) { ... }

# MONOCHROME

- Wert: <integer>
- Anwendbar für visuelle Medientypen.
- Kann mit min/max kombiniert werden.
- Die monochrome - Angabe bezeichnet die Anzahl der bits pro Pixel eines einfarbigen Bildschirmspuffers.
- Bei nicht monochromen Geräten wird 0 geliefert.

# ZWEI WEGE FÜR EIN MONOCHROMES GERÄT

- @media all and (monochrome) { ... }
- @media all and (min-monochrome: 1) { ... }

# RESOLUTION

- Wert: <resolution>
- Anwendbar für bitmap basierte Medien
- Kann mit min/max kombiniert werden.
- resolution beschreibt die Auflösung des Ausgabegerätes, zum Beispiel die Pixeldichte. Medien mit nicht-quadratischen Pixeln muss die least-dense mit dem Wert von min-resolution verglichen werden, max-resolution mit der most-dense Dimension. Ein einfache 'resolution' - Angabe funktioniert niemals mit einem Gerät mit nicht-quadratischen Pixeln.
- Bei Druckern korrespondiert die 'resolution' mit der Druckauflösung (Anzahl der Druckpunkte) bei beliebigen Farbmodell.

# BEISPIELE FÜR DRUCKER

- @media print and (min-resolution: 300dpi) { ... }
- @media print and (min-resolution: 118dpcm) { ... }

# SCAN

- Wert: progressive | interlace
- Anwendbar für 'tv' Medien
- Kann nicht mit min/max kombiniert werden.
- 'scan' beschreibt das Zeilenaufbauverfahren eines TV-Ausgabegerätes.

# EIN BEISPIEL FÜR TV GERÄTE MIT PROGRESSIVEN SCANNING (ZEILENAUFBAU)

- @media tv and (scan: progressive) { ... }

# GRID

- Wert: <integer>
- Anwendbar für visuelle und taktile Medien.
- Kann nicht mit min/max kombiniert werden.
- 'grid' wird verwendet, um Geräte mit einer festen Bildschirmmatrix (grid) zu verwenden. Das kann ein 'tty'-Terminal sein oder auch ein Handydisplay mit einer festgelegten Zeichenanzahl- und Größe. 'grid' liefert bei solchen Geräten eine '1'. Bitmapbasierte Displays erzeugen eine '0'.
- Andere Werte als '0' oder '1' erzeugen ein falsches Mediaquery.

ZWEI BEISPIELE:

1EM ENTSPRICHT DABEI EINER ZEICHENBREITE  
ODER -HÖHE.

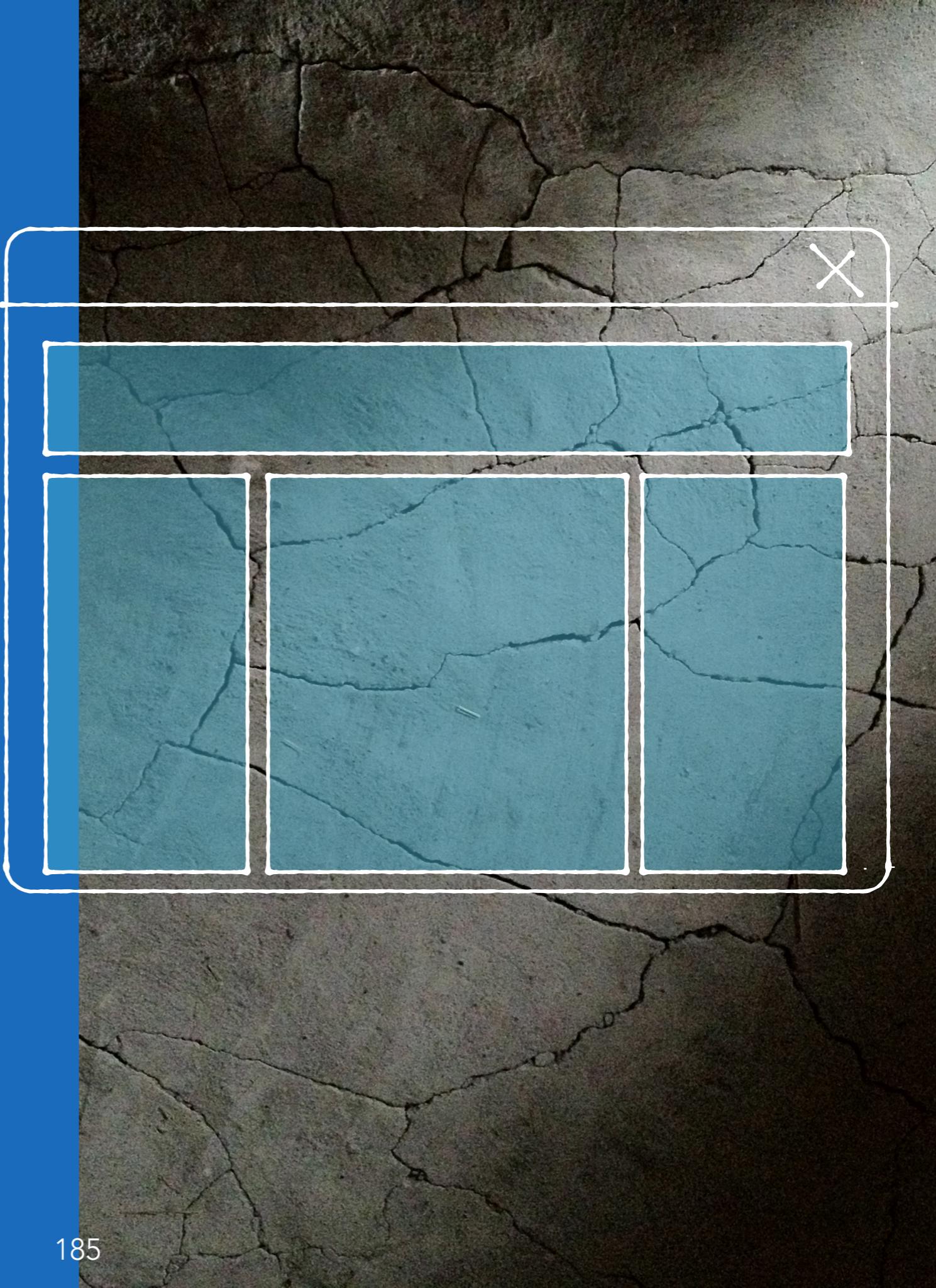
- @media  
handheld  
and (grid)  
and (max-width: 15em) { ... }
- @media  
handheld  
and (grid)  
and (device-max-height: 7em) { ... }

FÜR MOBILE GERÄTE HEUTE STANDARD UND AUSREICHEND.  
AUßerdem: MOBILE FIRST - CONTENT FIRST.

ADAPTIVES DESIGN.

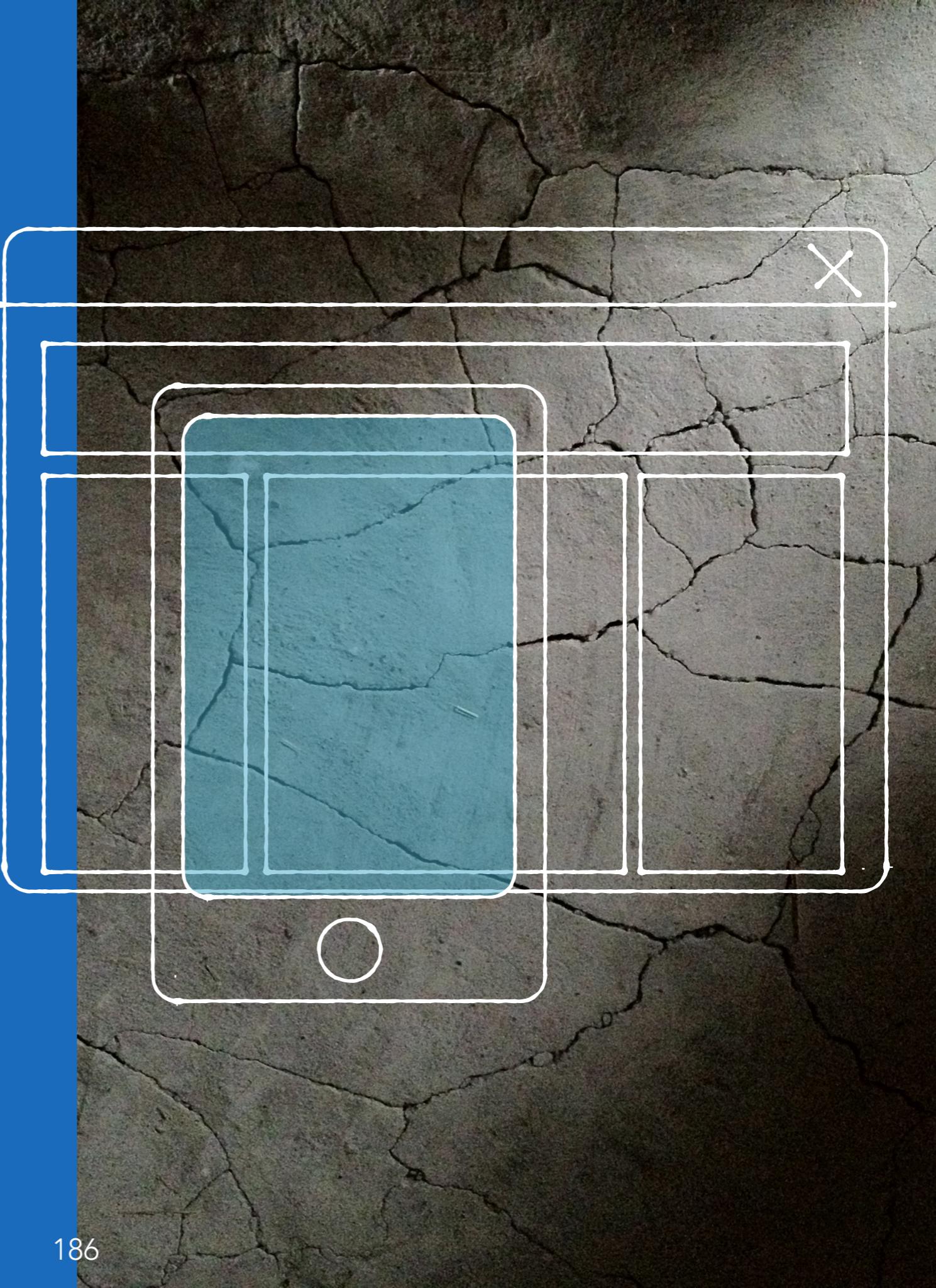
# DESKTOP-LAYOUTS SIND MEHRSPALTIG.

- Es gibt genügend Raum, um Navigation und Inhalte nebeneinander zu platzieren.



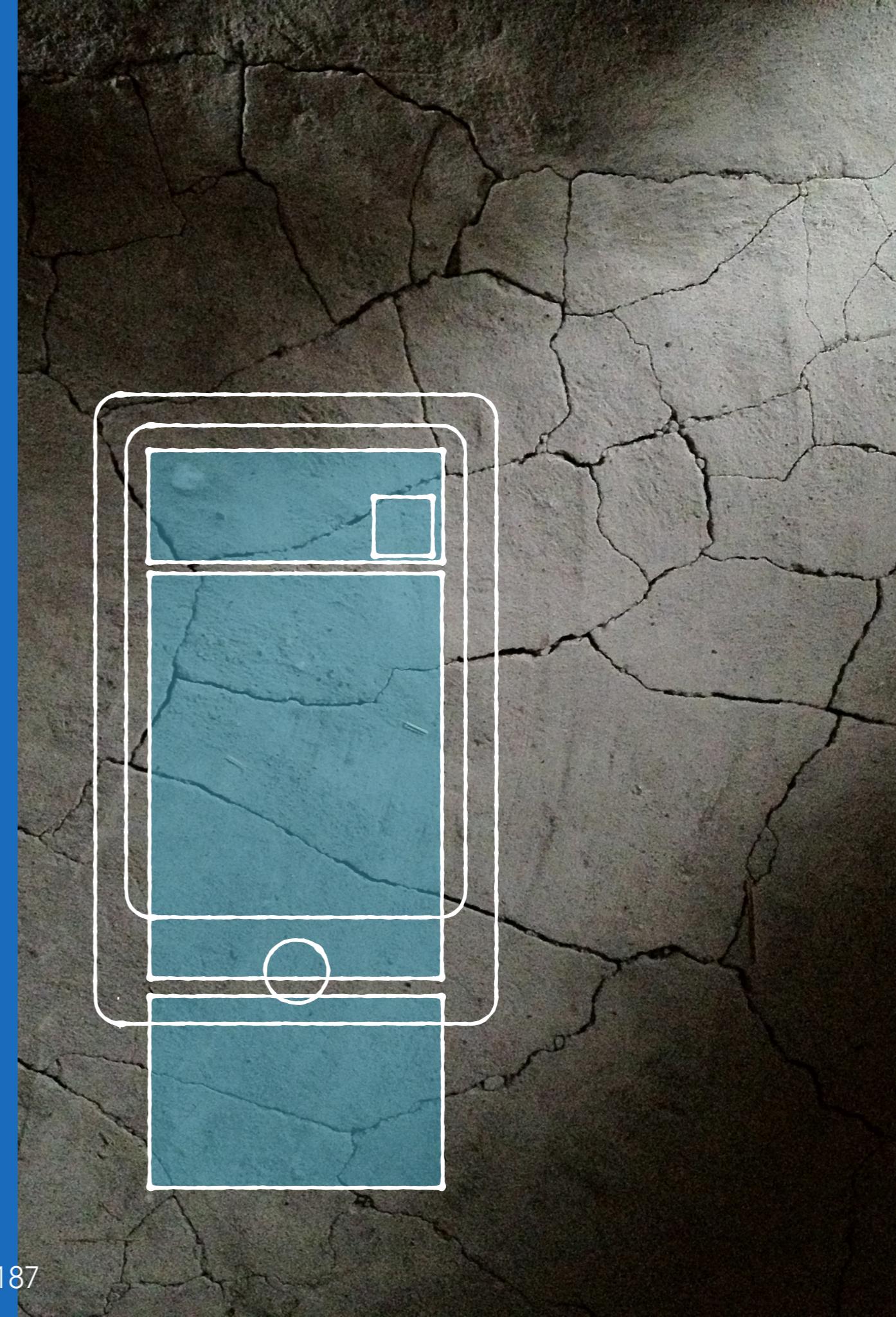
# SMARTPHONE LAYOUTS NICHT.

- Besonders im Hochformat biete das Layout oft nur die Hälfte oder ein Drittel des Raumes.
- Die Anordnung der Layoutelemente muss geändert werden.



# LINEARISIERUNG DER INHALTE.

- Die Inhalte werden untereinander angeordnet, (Flow), einige Elemente werden gar nicht angezeigt.
- Mobile First = Content First



# LINEARISIERUNG DES GRIDS MIT MEDIA QUERIES.

Eine einfache Ergänzung: das Grid wird linearisiert, jede Spalte liegt unter der anderen.

```
@media only screen  
  and (max-width: 767px) {  
  ...  
}
```

(Leider wird das bis IE8 nicht funktionieren.)

# ÜBERSCHREIBEN DER WIDTH UND MARGIN/PADDING ATTRIBUTE

```
.row {  
width: auto;  
min-width: 0;  
max-width: 100%;  
padding: 0 15px;  
margin-left: 0;  
margin-right: 0;  
}  
.row .span3 {width:25%;} muss überschrieben  
werden!  
.column, .columns {  
width: auto !important;  
float: none;  
}
```

# FLOATS ZURÜCKNEHMEN.

```
.column:last-child, .columns:last-child {  
  float: none;  
}  
  
[class*="column"]  
+ [class*="column"]:last-child {  
  float: none;  
}  
  
.column:after, .columns:after {  
  clear: both;  
}
```

# FÜR ALLE DISPLAYBREITEN EINE PASSENDE VARIANTE.

	Extra small devices Phones (<768px)	Small devices Tablets (≥768px)	Medium devices Desktops (≥992px)	Large devices Desktops (≥1200px)
Grid behavior	Horizontal at all times	Collapsed to start, horizontal above breakpoints		
Max container width	None (auto)	750px	970px	1170px
Class prefix	.col-xs-	.col-sm-	.col-md	.col-lg
# of columns	12			
Max column width	Auto	60px	78px	95px
Gutter width	30px (15px on each side of a column)			
Nestable				
Offsets	N/A	Yes		
Column ordering	N/A	Yes		

# DIE MEDIAQUERIES AUS BOOTSTRAP

```
/* Extra small devices (phones, less than 768px) */  
/* No media query since this is the default in Bootstrap */  
  
/* Small devices (tablets, 768px and up) */  
@media (min-width: @screen-sm-min) { ... }  
  
/* Medium devices (desktops, 992px and up) */  
@media (min-width: @screen-md-min) { ... }  
  
/* Large devices (large desktops, 1200px and up) */  
@media (min-width: @screen-lg-min) { ... }
```

# RESPOND.JS FÜR IE 6-8

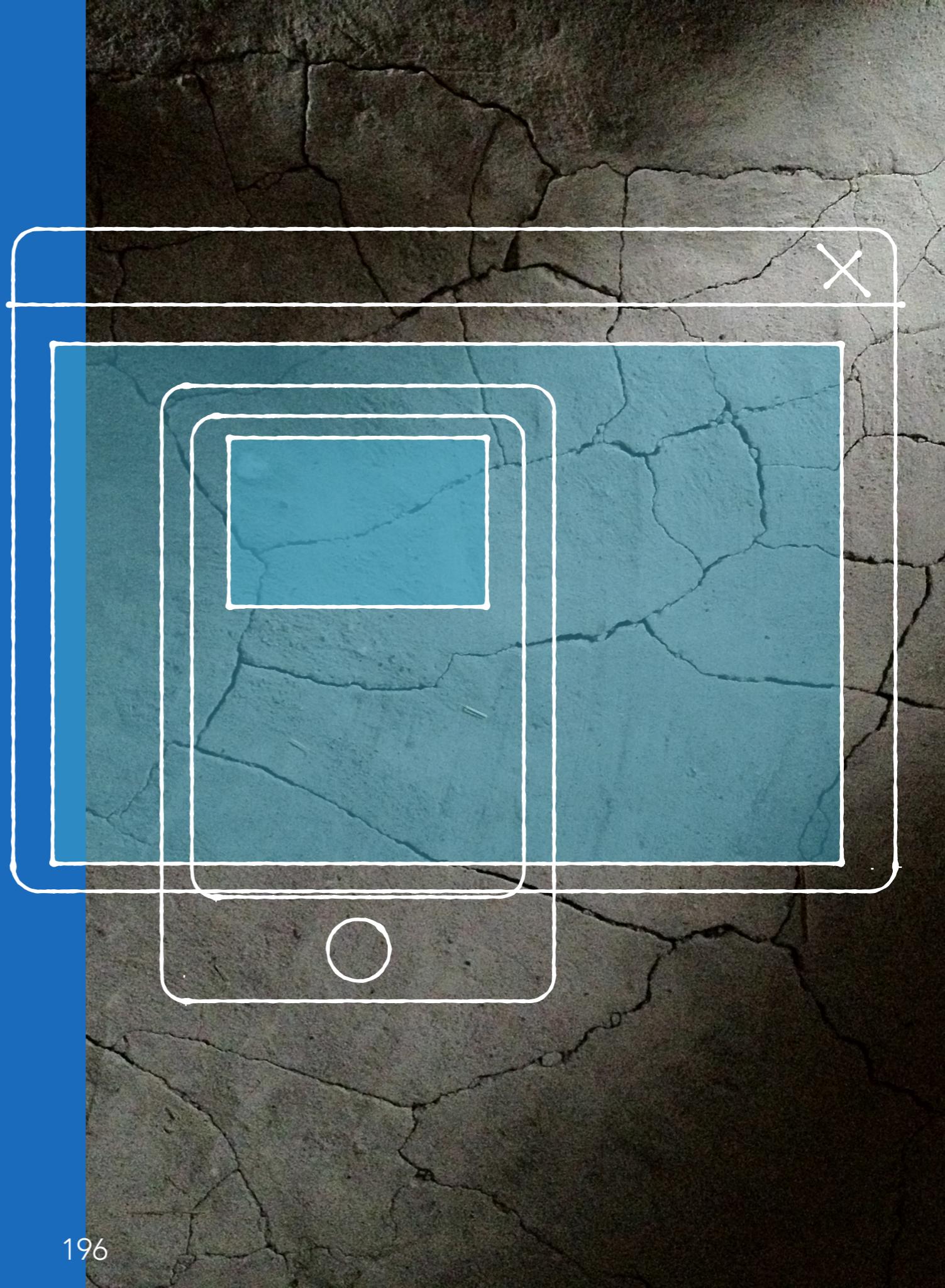
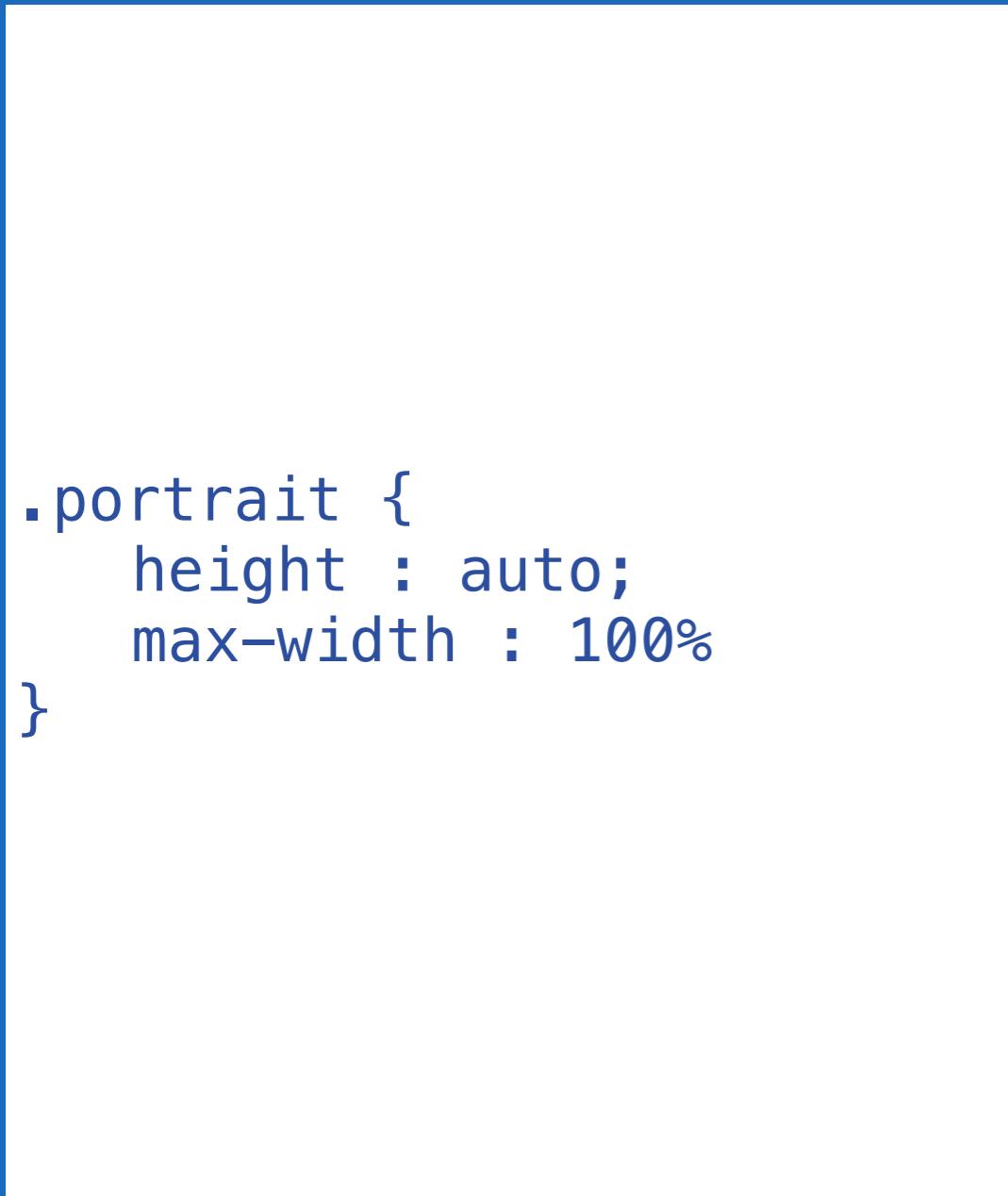
- A fast & lightweight polyfill for min/max-width CSS3 Media Queries (for IE 6-8, and more).
- Respond.js bildet die fehlenden min- und max-width Attribute für Internet Explorer 6 bis 8 nach. Damit sind die adaptiven und fluiden Layouts auch dort anwendbar.

<https://github.com/scottjehl/Respond>

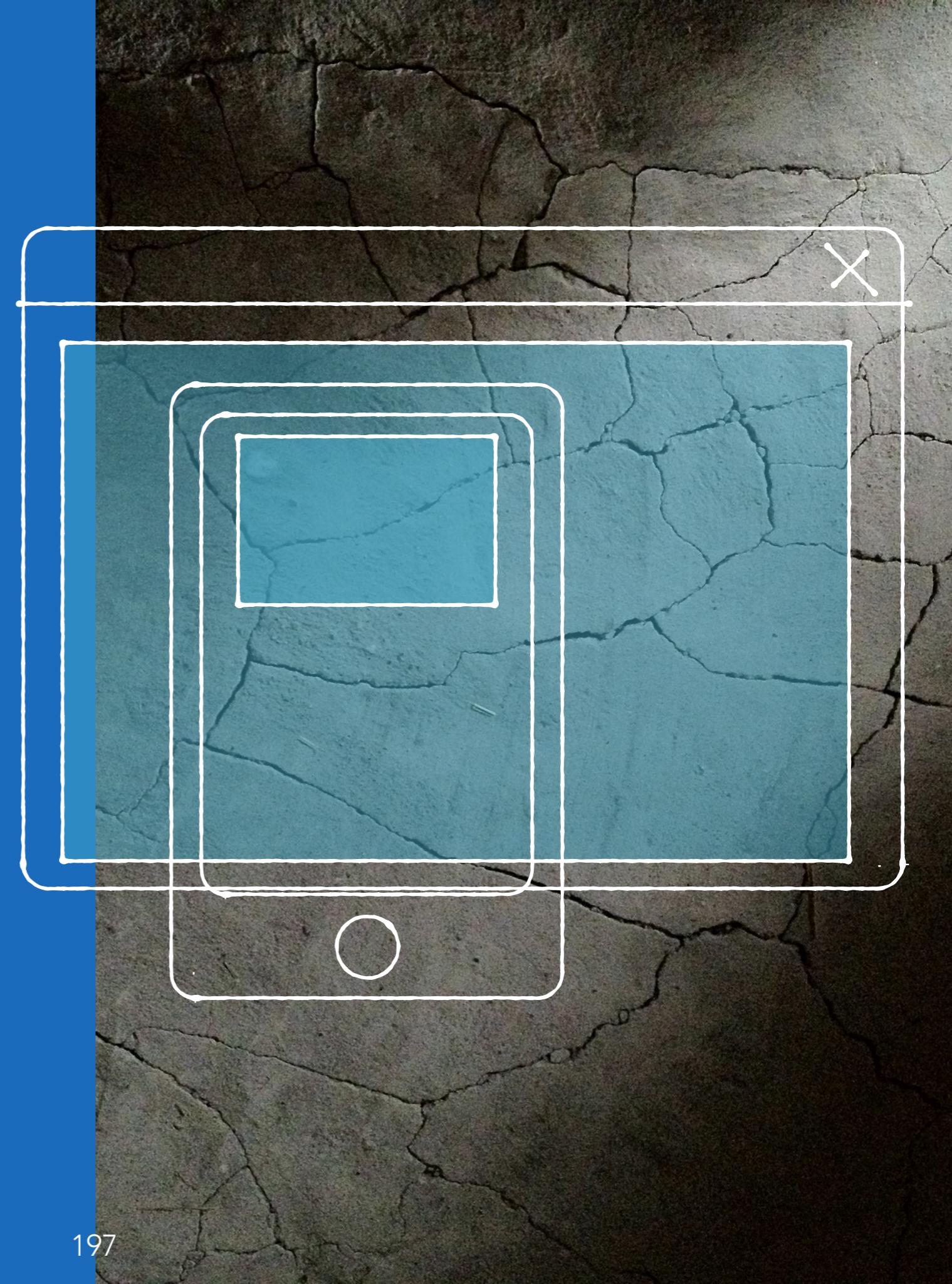
- SCOTT JEH'L

# RESPONSIVE BILDER

# BILDER RESPONSIVE BEMASSEN

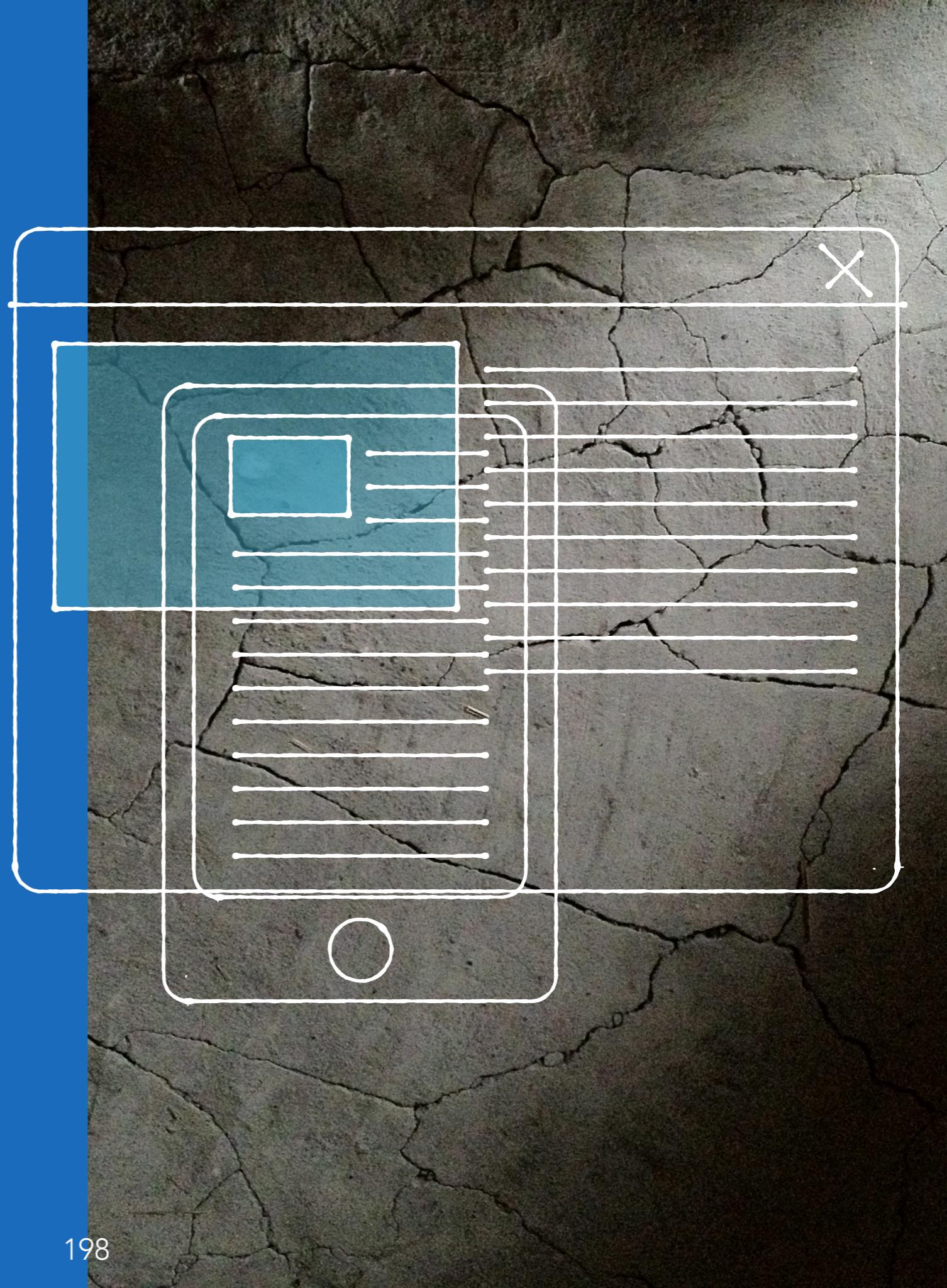
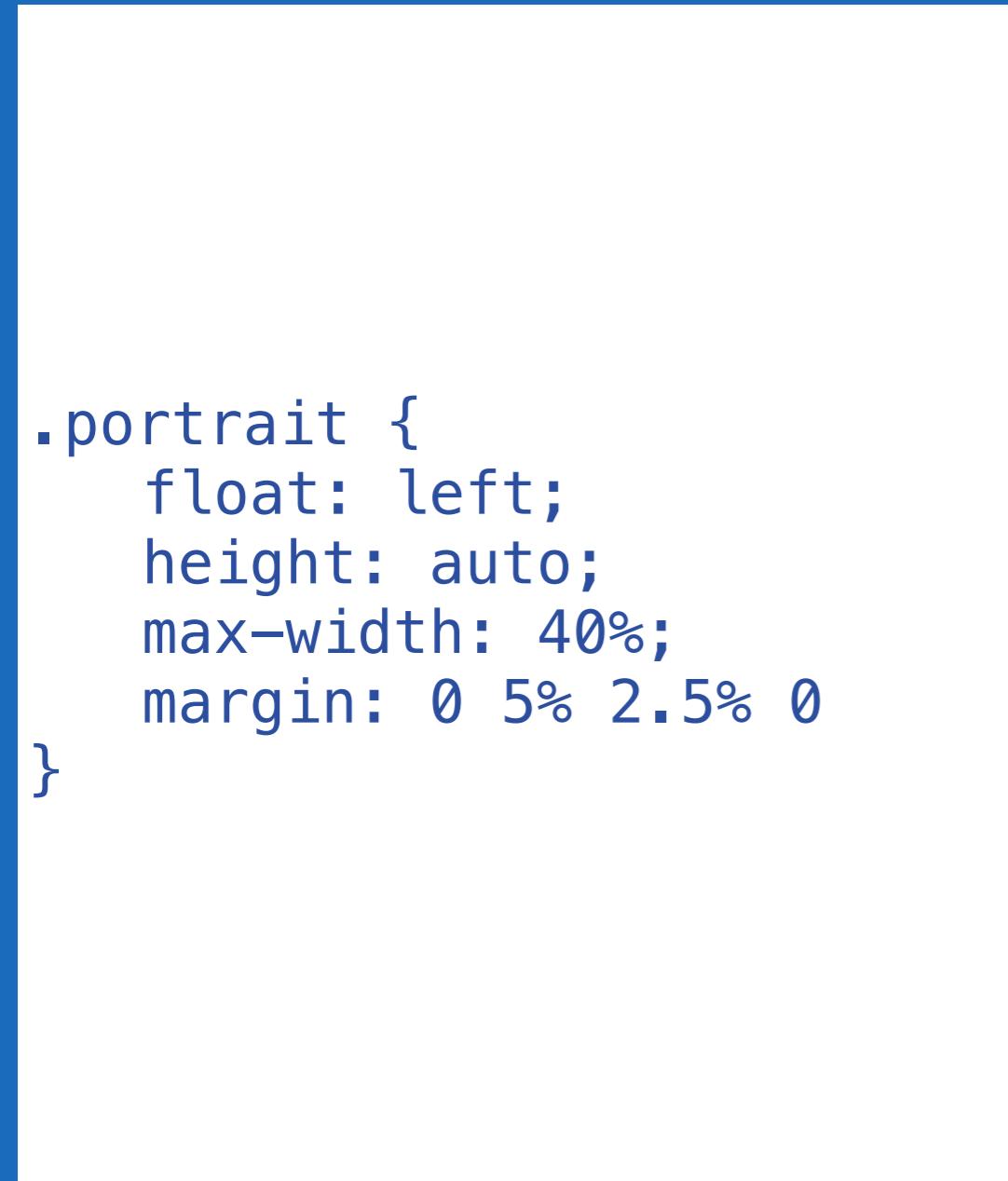


Das geht auch mit  
embed, object, video!



# BILDER RESPONSIVE BEMASSEN

```
.portrait {  
    float: left;  
    height: auto;  
    max-width: 40%;  
    margin: 0 5% 2.5% 0  
}
```



# AUFLÖSUNGEN BEACHTEN - DIE GRENZEN BEIM SKALIEREN VON BILDERN

- Bilder lassen sich allerdings weder vergrößern, noch beliebig verkleinern. Ein Pixel ist ein Pixel, und wird davon nur die Hälfte gezeigt, geht Qualität verloren.

# DAS <PICTURE> ELEMENT

Für die Platzierung von Bildern wird ein neues <picture>-Element eingeführt. Damit können auflösungsabhängig verschiedene Bildformate bereitgestellt werden. Dabei müssen die Bilder vom größten zum kleinsten hin eingebunden werden.

```
<picture>
  <source media="(min-width: 1200px)" srcset="1440px.jpg">
  <source media="(min-width: 992px)" srcset="1199px.jpg">
  <source media="(min-width: 768px)" srcset="991px.jpg" v>
  <source media="(min-width: 481px)" srcset="767px.jpg" v>
  <source media="(max-width: 480px)" srcset="480px.jpg" v>
  
</picture>
```

# DAS SRCSET ATTRIBUT

Die srcset Attribute halten verschiedenen Bildquellen bereit, von denen eine nach der Auswertung des Browsers ausgewählt wird und anstelle der src Angabe im img Element verwendet wird.

```
<picture>
  <source media="(min-width: 1200px)" srcset="1440px.jpg">
  <source media="(min-width: 992px)" srcset="1199px.jpg">
  <source media="(min-width: 768px)" srcset="991px.jpg" v v v
  <source media="(min-width: 481px)" srcset="767px.jpg" v v v
  <source media="(max-width: 480px)" srcset="480px.jpg" v v v
  
</picture>
```

# DAS FALBACK BILD

Trifft keine der srcset Bilder zu oder kann ein Browser das picture/srcset nicht auswerten, dann wird das img Element als Fallback verwendet.

```
<picture>
  <source media="(min-width: 1200px)" srcset="1440px.jpg">
  <source media="(min-width: 992px)" srcset="1199px.jpg">
  <source media="(min-width: 768px)" srcset="991px.jpg" >
  <source media="(min-width: 481px)" srcset="767px.jpg" >
  <source media="(max-width: 480px)" srcset="480px.jpg" >
  
</picture>
```

# <PICTURE>

Ausser der gerätespezifischen Einstellung kann das <picture> Element auch zur Auswahl aus mehreren alternativen Bildformaten verwendet werden.

```
<picture>
  <source srcset="/uploads/100-marie-lloyd.webp"
          type="image/webp">
  <source srcset="/uploads/100-marie-lloyd.jxr"
          type="image/vnd.ms-photo">
  
</picture>
```

# RETINA DISPLAYS FÜR <PICTURE>

```
<picture>
  <source media="(min-width: 1200px),
              (-webkit-min-device-pixel-ratio: 2),
              (min-resolution: 192dpi)" srcset="2880px.jpg">
  <source media="(min-width: 1200px)" srcset="1440px.jpg">
  <source media="(min-width: 992px)" srcset="1199px.jpg">
  <source media="(min-width: 768px)" srcset="991px.jpg" >
  <source media="(min-width: 481px)" srcset="767px.jpg" >
  <source media="(max-width: 480px)" srcset="480px.jpg" >
  
</picture>

@media
(-webkit-min-device-pixel-ratio: 2),
(min-resolution: 192dpi) {
  /* Retina-specific stuff here */
}
```

<https://css-tricks.com/snippets/css/retina-display-media-query/>

# <IMG SRC="..." SRCSET="..."> SRCSET-W

Ebenfalls für auflösungsabhängig Bildformate kann das `srcset`-Attribut verwendet werden. Im Gegensatz zur `picture` Gruppe wertet das `srcset`-Attribut browserseitig und selbstständig aus, welche Bilddatei unter gegebenen Bedingungen die beste ist.

`srcset-w:`

```

```

```
<IMG SRC="..." SRCSET="...">  
SRCSET-X
```

Mit der x-Angabe können Bilder Display-auflösungsabhängig hinterlegt werden. So lassen sich Retina-fähige Bilder hinterlegen.

```
srcset-x:  

```

[https://css-tricks.com/  
responsive-images-youre-just-changing-resolutions-  
use-srcset/](https://css-tricks.com/responsive-images-youre-just-changing-resolutions-use-srcset/)

[https://www.smashingmagazine.com/  
2014/05/responsive-images-done-right-guide-  
picture-srcset/](https://www.smashingmagazine.com/2014/05/responsive-images-done-right-guide-picture-srcset/)

# PICTUREFILL

## A RESPONSIVE IMAGE POLYFILL

- The picture element and associated features are W3C standard HTML features that allow web developers to deliver an appropriate image to every user depending on a variety of conditions like screen size, viewport size, screen resolution, and more.
- Picturefill is a JavaScript file (or a polyfill to be more specific) that enables support for the picture element and associated features in browsers that do not yet support them, so you can start using them today!

<http://scottjehl.github.io/picturefill/>

# SERVERSIDE: ADAPTIVE- IMAGES.PHP

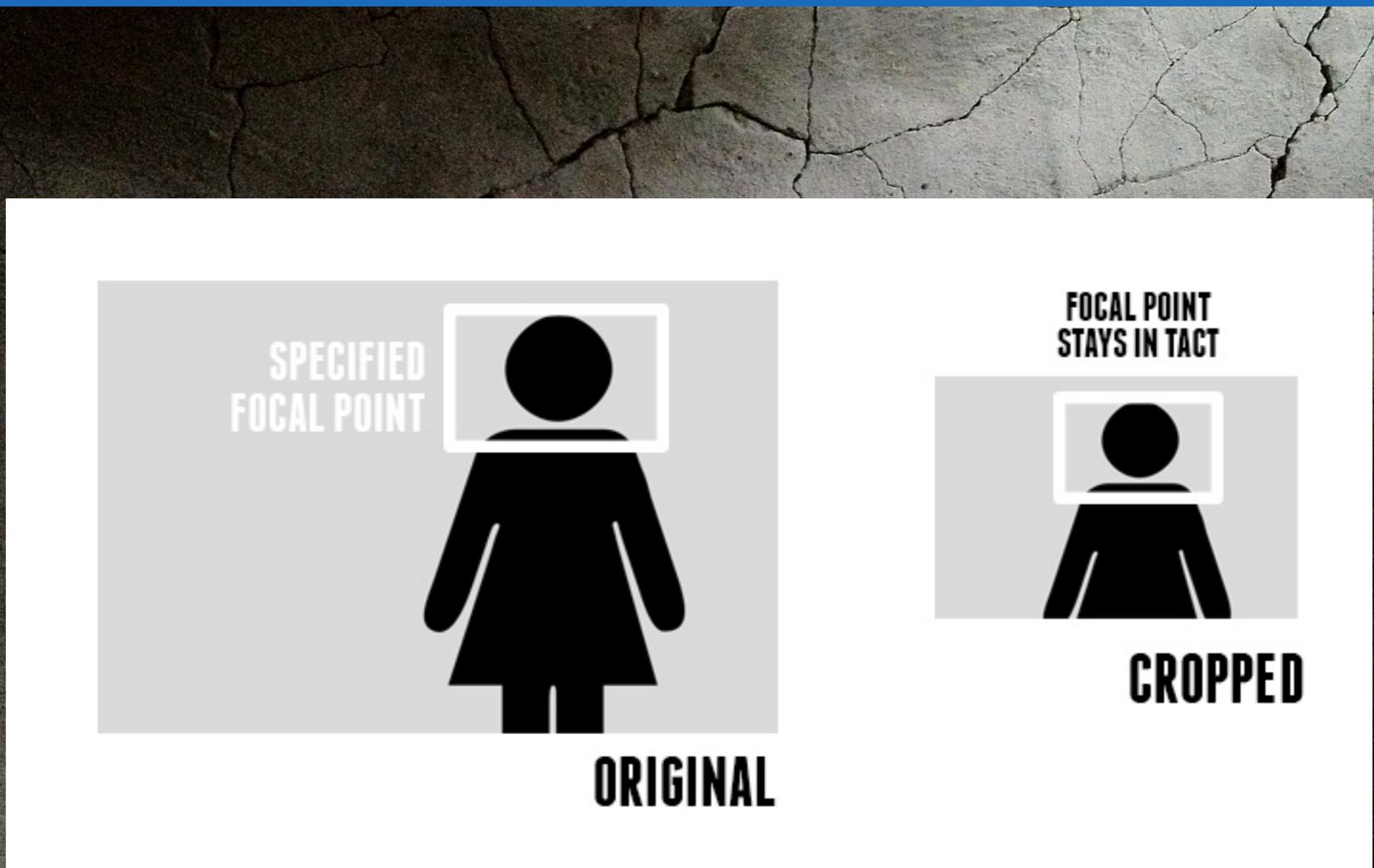
- Adaptive Images detects your visitor's screen size and automatically creates, caches, and delivers device appropriate re-scaled versions of your web page's embedded HTML images. No mark-up changes needed. It is intended for use with Responsive Designs and to be combined with Fluid Image techniques.

<http://www.adaptive-images.com>

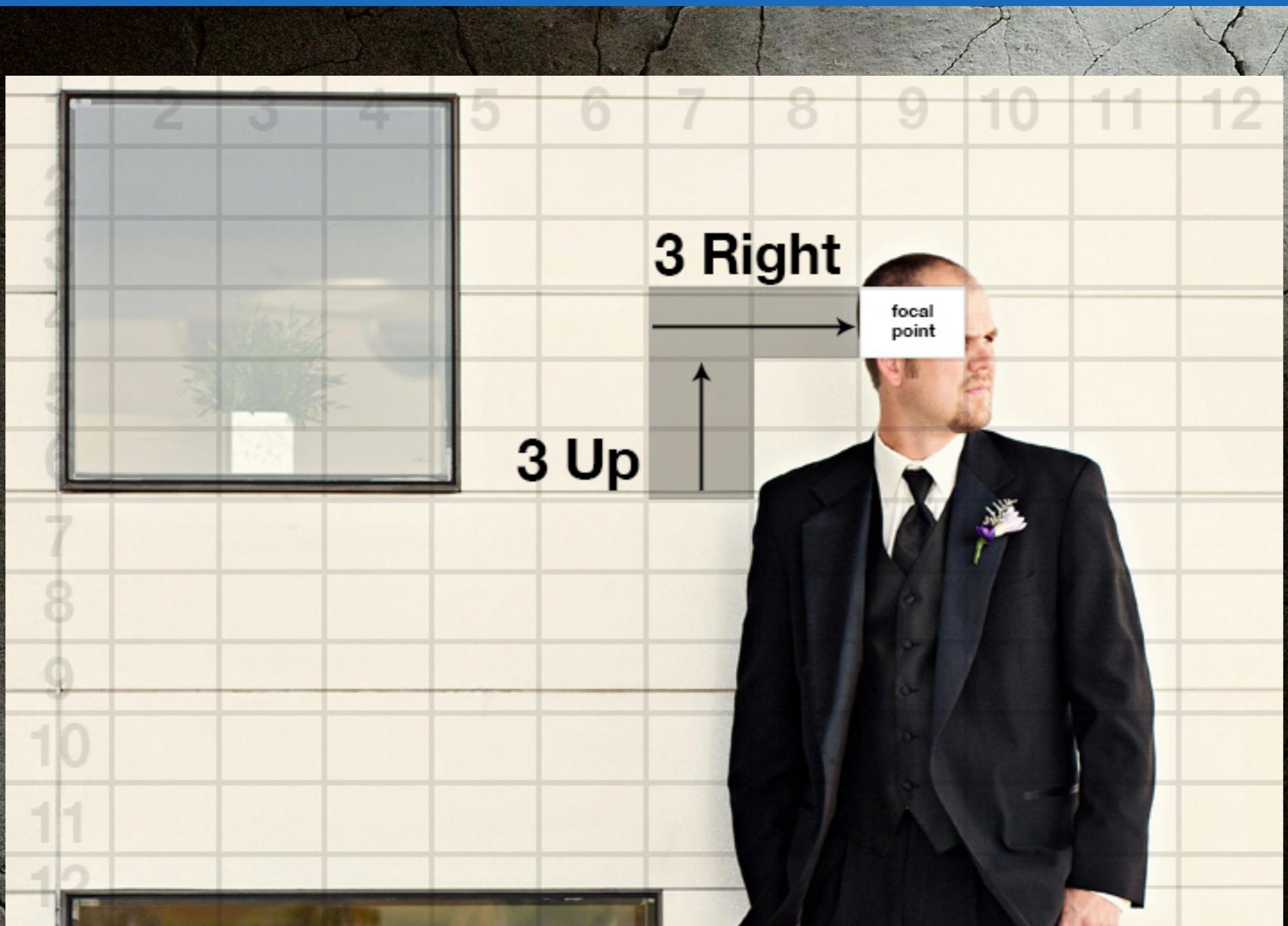
# FOCAL POINT: INTELLIGENT CROPPING OF RESPONSIVE IMAGES



# FOCAL POINT: INTELLIGENT CROPPING OF RESPONSIVE IMAGES



# FOCAL POINT: INTELLIGENT CROPPING OF RESPONSIVE IMAGES



# FOCAL POINT: INTELLIGENT CROPPING OF RESPONSIVE IMAGES

```
<div class="focal-point right-3 up-3">  
  <div></div>  
</div>
```

# FOCAL POINT: INTELLIGENT CROPPING OF RESPONSIVE IMAGES

```
.focal-point {  
    width      : 100%;  
    height     : auto;  
    overflow   : hidden;  
}  
.focal-point img {  
    width      : 100%;  
    max-width  : 100%;  
    height     : auto;  
    -ms-interpolation-mode : bicubic;  
}  
.focal-point div {  
    position   : relative;  
    max-width  : none;  
    height     : auto;  
}
```

# FOCAL POINT: INTELLIGENT CROPPING OF RESPONSIVE IMAGES

```
@media all and (max-width: 767px) {  
  /* 4x3 Landscape Shape (Default) */  
  .focal-point div {  
    margin           : -3em -4em;  
  }  
  /* Landscape up (Total 6em) */  
  .up-3 div {  
    margin-top      : -1.5em;  
    margin-bottom   : -4.5em;  
  }  
  .right-3 div {  
    margin-left     : -6em;  
    margin-right    : -2em;  
  }  
}
```

<http://designshack.net/articles/css/focal-point-intelligent-cropping-of-responsive-images/>

# GESTALTUNG VON INHALTEN CONTENT

# SCHRIFTEN

# SCHRIFTEINBINDUNG

# EINBINDUNG VON ZEICHENSÄTZEN

```
@font-face {  
    font-family: 'LeagueGothic';  
    src: url(LeagueGothic.otf);  
}  
  
header {  
    font-family: 'LeagueGothic', Arial;  
}  
  
@font-face {  
    font-family: 'Droid Sans';  
    src: url(Droid_Sans.ttf);  
}  
@font-face {  
    font-family: 'Droid Sans';  
    src: url(Droid_Sans.woff);  
}
```

# EINBINDEN VERSCHIEDENER FORMATE

```
@import url(http://fonts.googleapis.com/css?  
family=Open+Sans:300italic,400italic,600italic,  
700italic,800italic,400,300,600,700,800);  
@font-face {  
    font-family : OpenSans;  
  
src : url("../libraries/fonts/Open_Sans/OpenSans-Light.ttf") format("truetype");  
font-weight : 100;  
font-weight : 300;  
src : url("../libraries/fonts/Open_Sans/OpenSans-Regular.ttf");  
font-weight : 400;  
src : url("../libraries/fonts/Open_Sans/OpenSans-Semibold.ttf");  
font-weight : 600;  
}
```

# GLYPHICONS - SCHRIFT ALS ICONSET NUTZEN

```
@font-face {  
    font-family: 'Glyphicons Halflings';  
  
    src: url('../fonts/glyphicons-halflings-regular.eot');  
    src: url('../fonts/glyphicons-halflings-regular.eot?#iefix')  
        format('embedded-opentype'),  
  
    url('../fonts/glyphicons-halflings-regular.woff')  
        format('woff'),  
  
    url('../fonts/glyphicons-halflings-regular.ttf')  
        format('truetype'),  
  
    url('../fonts/glyphicons-halflings-regular.svg#glyphicons_halflingsregular')  
        format('svg');  
}  
.glyphicon {  
    position: relative;  
    top: 1px;  
    display: inline-block;  
    font-family: 'Glyphicons Halflings';  
    ...  
}
```

# GLYPHICONS - SCHRIFT ALS ICONSET NUTZEN

```
.glyphicon-asterisk:before {  
    content: "\2a";  
}  
.glyphicon-plus:before {  
    content: "\2b";  
}  
...  
...
```

# DIE GRÖÙE VON ZEICHENSÄTZEN JUSTIEREN

```
p.beispiel {  
    font-family:  
    Verdana, Arial, sans-serif;  
    font-size:12px;  
    font-size-adjust:0.58;  
}
```

Mit `font-size-adjust` kann man eine bestimmte x-Höhe erzwingen, ganz egal welche Schrift aus `font-family` eingesetzt wird. Im Beispiel wird die Schriftliste auf den Adjustwert der Verdana eingestellt.

Siehe auch

<http://www.webspaceworks.com/resources/fonts-web-typography/43/>

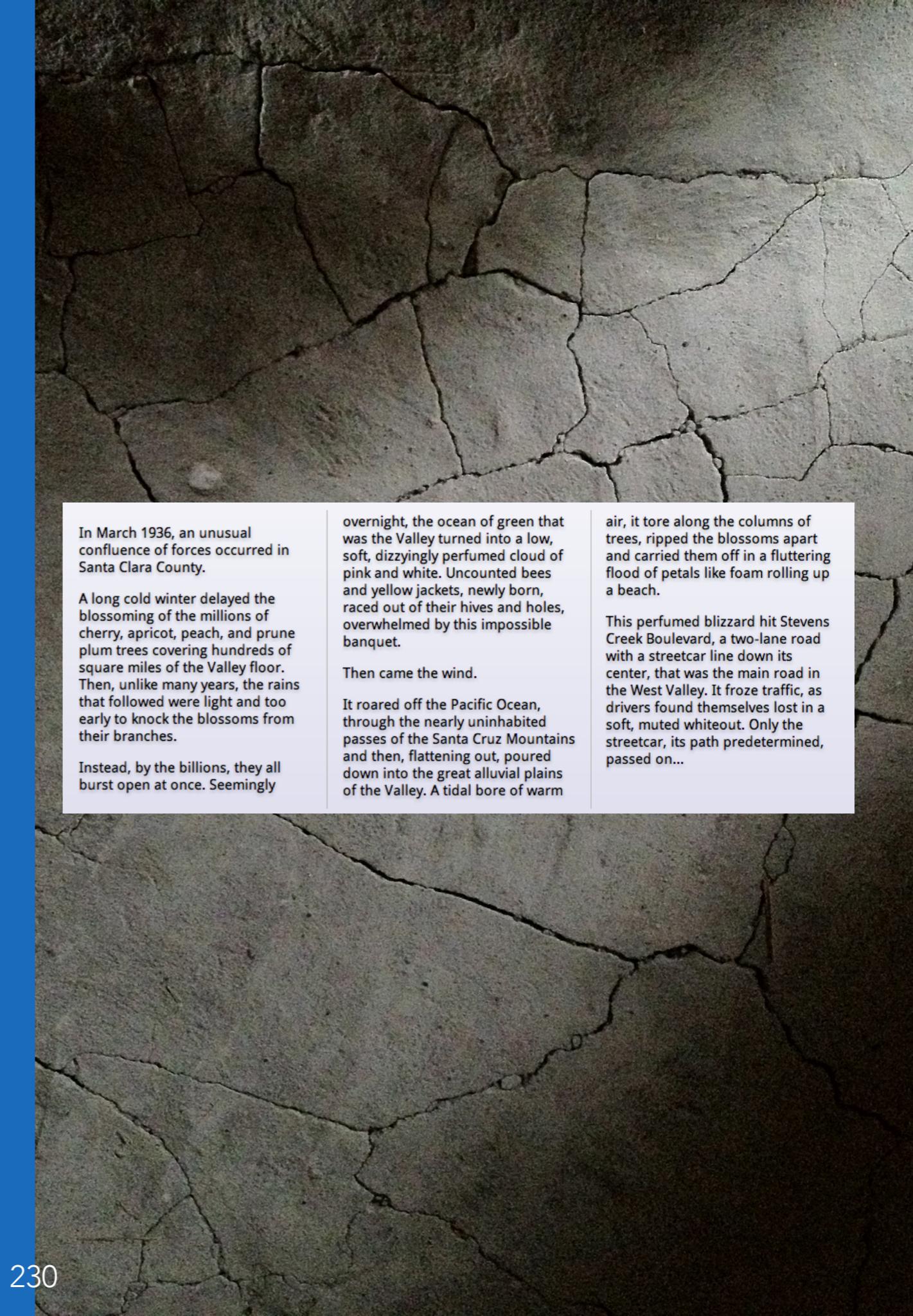
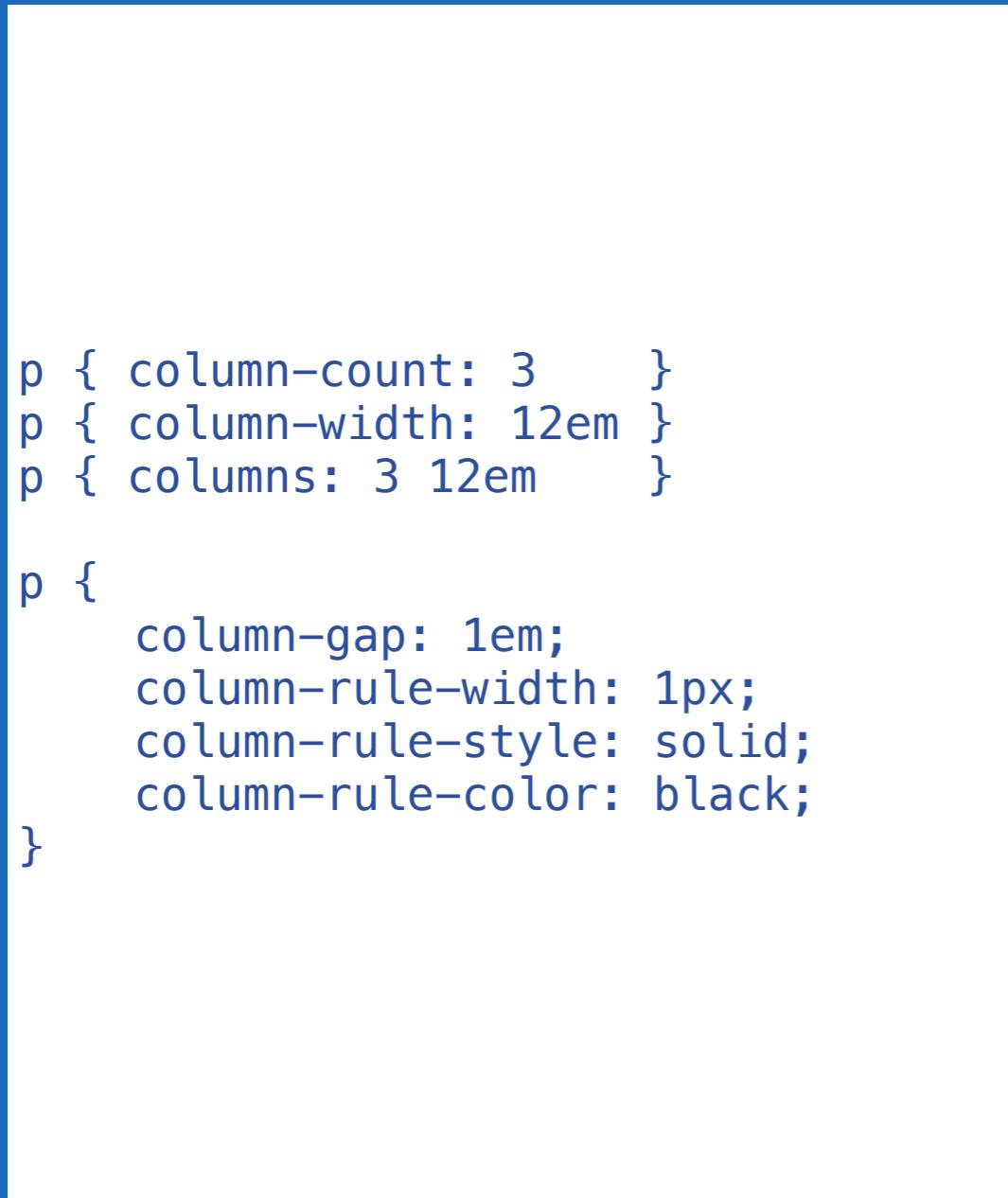
# TEXTKONTUREN

```
div {  
  outline:  
    text-fill-color: black;  
    text-stroke-color: red;  
    text-stroke-width: 4.00px;  
}
```

# MEHRSPALTENSATZ

# MEHRSPALTENSATZ - DAS MULTICOLUMN MODEL

```
p { column-count: 3 }  
p { column-width: 12em }  
p { columns: 3 12em }  
  
p {  
    column-gap: 1em;  
    column-rule-width: 1px;  
    column-rule-style: solid;  
    column-rule-color: black;  
}
```



In March 1936, an unusual confluence of forces occurred in Santa Clara County.

A long cold winter delayed the blossoming of the millions of cherry, apricot, peach, and prune plum trees covering hundreds of square miles of the Valley floor. Then, unlike many years, the rains that followed were light and too early to knock the blossoms from their branches.

Instead, by the billions, they all burst open at once. Seemingly

overnight, the ocean of green that was the Valley turned into a low, soft, dizzyingly perfumed cloud of pink and white. Uncounted bees and yellow jackets, newly born, raced out of their hives and holes, overwhelmed by this impossible banquet.

Then came the wind.

It roared off the Pacific Ocean, through the nearly uninhabited passes of the Santa Cruz Mountains and then, flattening out, poured down into the great alluvial plains of the Valley. A tidal bore of warm

air, it tore along the columns of trees, ripped the blossoms apart and carried them off in a fluttering flood of petals like foam rolling up a beach.

This perfumed blizzard hit Stevens Creek Boulevard, a two-lane road with a streetcar line down its center, that was the main road in the West Valley. It froze traffic, as drivers found themselves lost in a soft, muted whiteout. Only the streetcar, its path predetermined, passed on...

# UMBRUCHSTEUERUNG BEI MEHRSPALTENSATZ

```
h1 {  
    break-before: column;  
    break-inside: avoid-column;  
    break-after: avoid-column;  
}  
  
div {  
    width: 100px;  
    column-width: 45px;  
    column-gap: 0;  
    column-rule: none;  
}  
  
h3 {  
    column-span: all; /* none | initial | inherit | unset */  
}
```

# SILBENTRENNUNG

```
p {  
    hyphens: none | manual | auto;  
}
```

manual → &shy;

none → keine Trennungen

auto → trennt immer dort, wo getrennt werden muss.

Ggf. Vendor-Präfixe, also „-moz-hyphens“, „-webkit-hyphens“

lang-Attribut verwenden, damit die Trennung nach Sprachregeln funktioniert!

# ERZWUNGENER WORTUMBRUCH

Für Wörter mit Überlänge und ohne Trennregel:

```
p {  
  word-wrap: normal | break-word;  
}
```

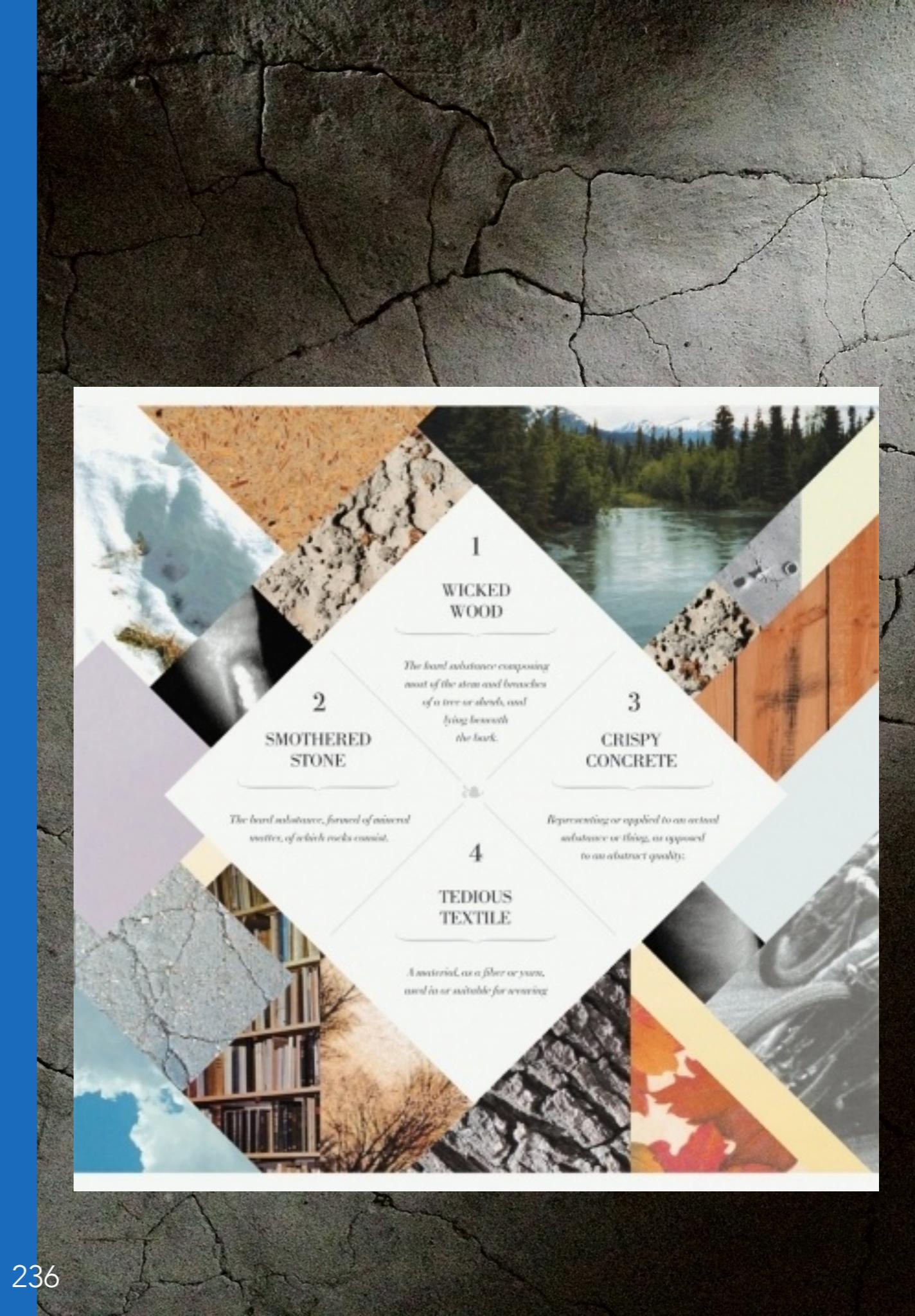
XML&shy;Http<wbr>Request

# SATZREGELN

```
p {  
    text-align : justify;  
    hyphens   : auto;  
    widows    : 3;          /* Schusterjungen-Regel */  
    orphans    : 3;          /* Hurenkinder-Regel */  
    word-spacing : normal;  
    letter-spacing : normal;  
    word-break  : normal;  
}
```

# FORMSATZ

# FORMSATZ MIT SHAPE-INSIDE



# FARBEN UND FARBVERLÄUFE, IMAGE PROCESSING

# RESPONSIVES LOGO

```
<div id="logo"><p class="sr-only">Company Name</p></div>

#logo {
    width      : 100%;
    height     : 100%;
    background : url(logo.svg) center center no-repeat;
    background-size : contain;
}
.sr-only {
    display   : block;
    position  : absolute;
    left     : -10000px
}
```

# RESPONSIVER VOLLBILD HINTERGRUND

```
body {  
    background : url(background.jpg) center center no-repeat;  
    background-size : cover;  
}
```

# BACKGROUND STACKING

In CSS 3 können mehrere Hintergrundeigenschaften kombiniert werden. Damit können auch komplexe grafische Designs mit cachebaren URLs (oder dem Application Cache) realisiert werden.

```
div {  
    background :  
        rgba(200,0,0,0.25),  
        linear-gradient(  
            to bottom,  
            #1e5799 0%,  
            #2989d8 50%,  
            #207cca 51%,  
            transparent 100%  
        );  
        url(src/overlay.png) 10px center no-repeat,  
        url(src/background.png) 10px center repeat-x;  
}
```

# NEUE FARBMODELLE MIT TRANSPARENZ

Für Farben kann nun eine Deckkraft eingestellt werden. Der vierte Wert steht für einen Alpha-Kanal. Im Gegensatz zu `Opacity`, welches jeweils für das ganze Element auswirkt, können mit den Alpha Farben Fläche, Rand oder Text einzeln eingestellt werden.

```
color:  
rgba(255, 0, 0, 0.75);  
color:  
hsla(360, 100%, 50%, 0.75);
```

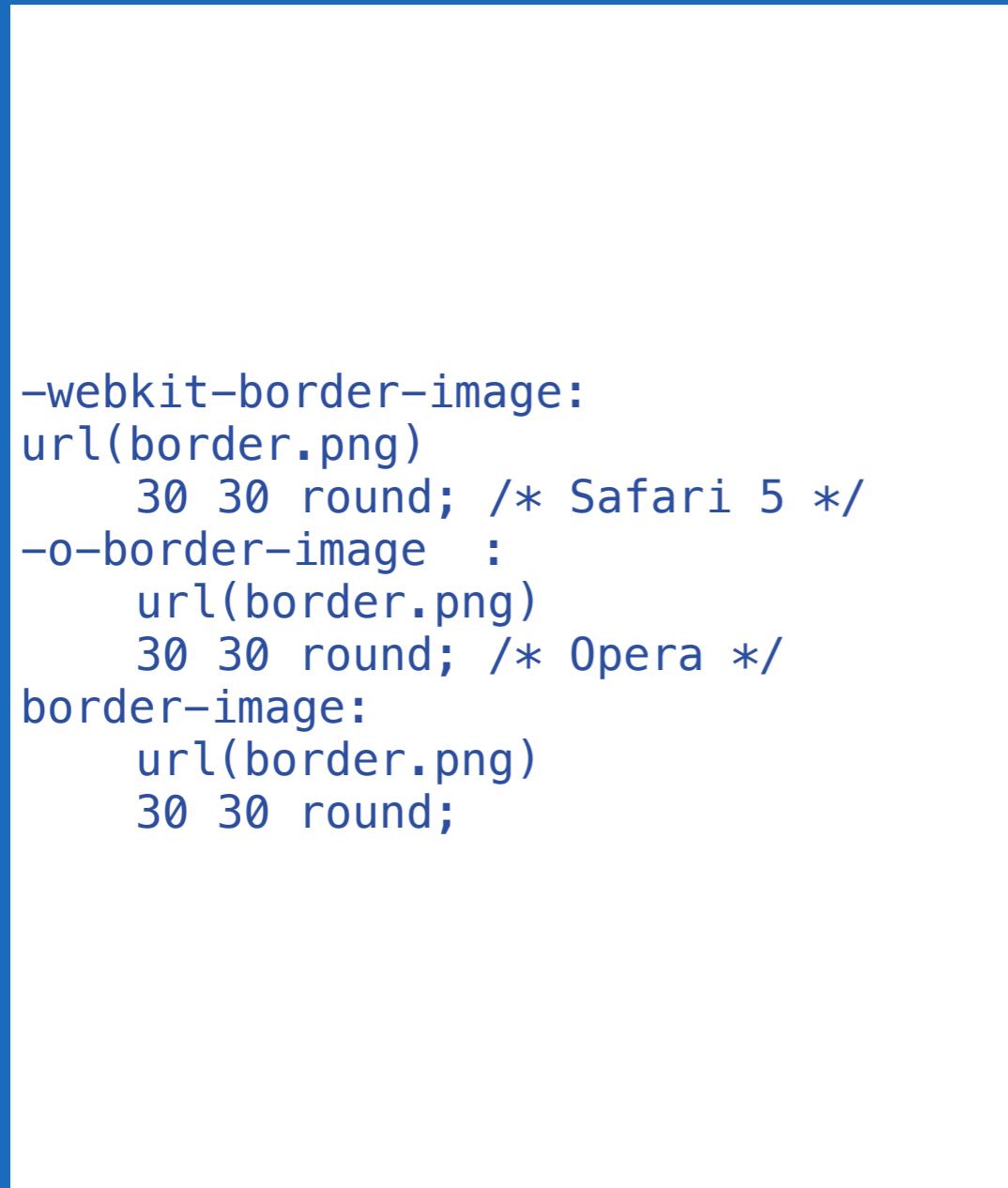
# FARBVERLÄUFE - LINEAR UND RADIAL

```
background:  
linear-gradient(  
    32deg,  
    #1e5799 0%,  
    #2989d8 50%,  
    #207cca 51%,  
    transparent 100%  
) ;
```

```
background:  
radial-gradient(  
    ellipse at center,  
    #1e5799 0%,  
    #2989d8 50%,  
    #207cca 51%,  
    #7db9e8 100%  
) ;
```

# CSS BORDER IMAGE

```
-webkit-border-image:  
url(border.png)  
30 30 round; /* Safari 5 */  
-o-border-image :  
url(border.png)  
30 30 round; /* Opera */  
border-image:  
url(border.png)  
30 30 round;
```



# ABGERUNDETE ECKEN FÜR ALLE BROWSER

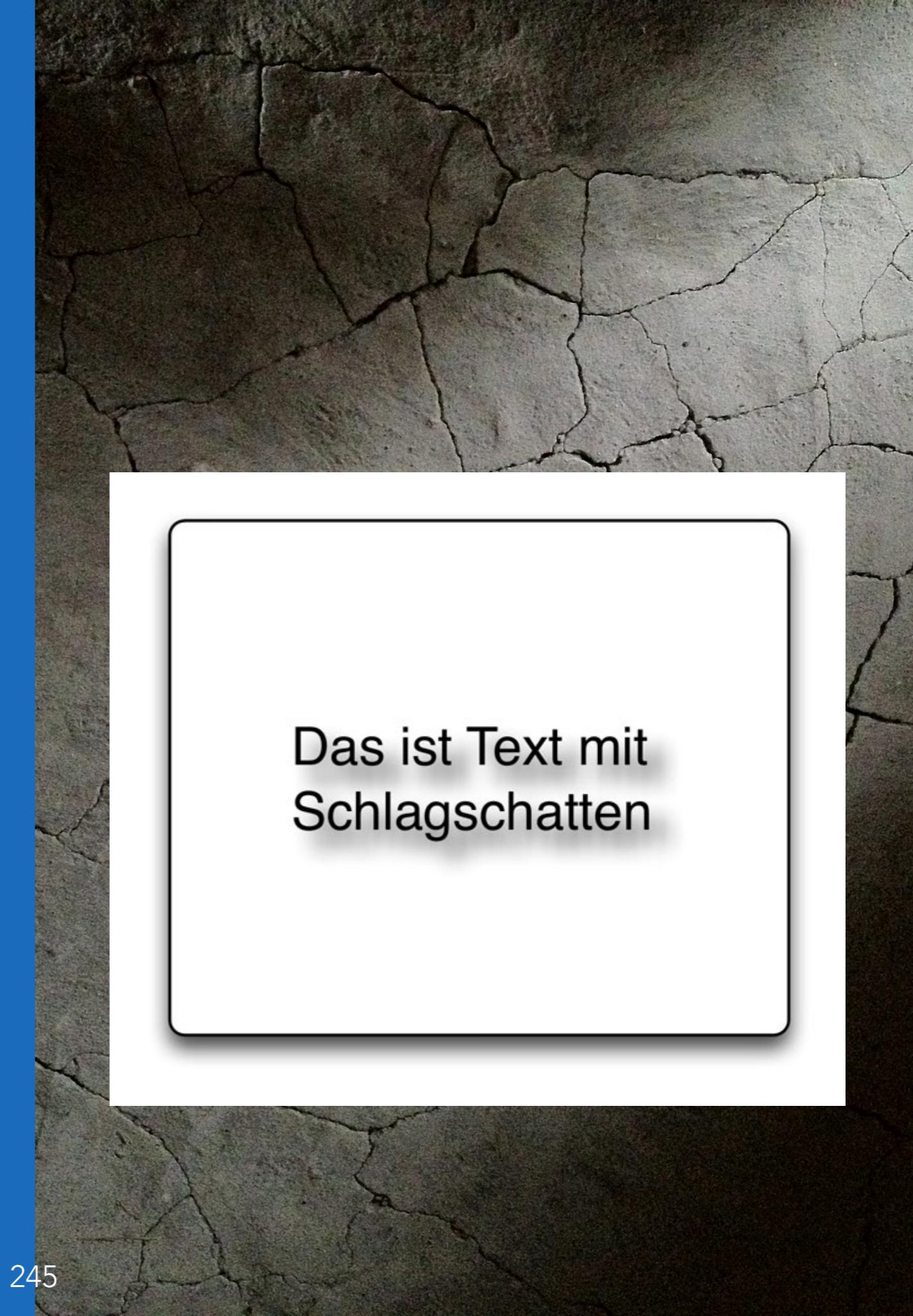
```
border-radius: 5px;  
border-left-top-radius:10px;
```

```
-moz-border-radius : 5px;  
-o-border-radius : 5px;  
-webkit-border-radius : 5px;  
-khtml-border-radius : 5px;  
-ms-border-radius : 5px;
```

Für alte IE's  
border-radius.htc  
(Google Coderepository)

# SCHATTEN

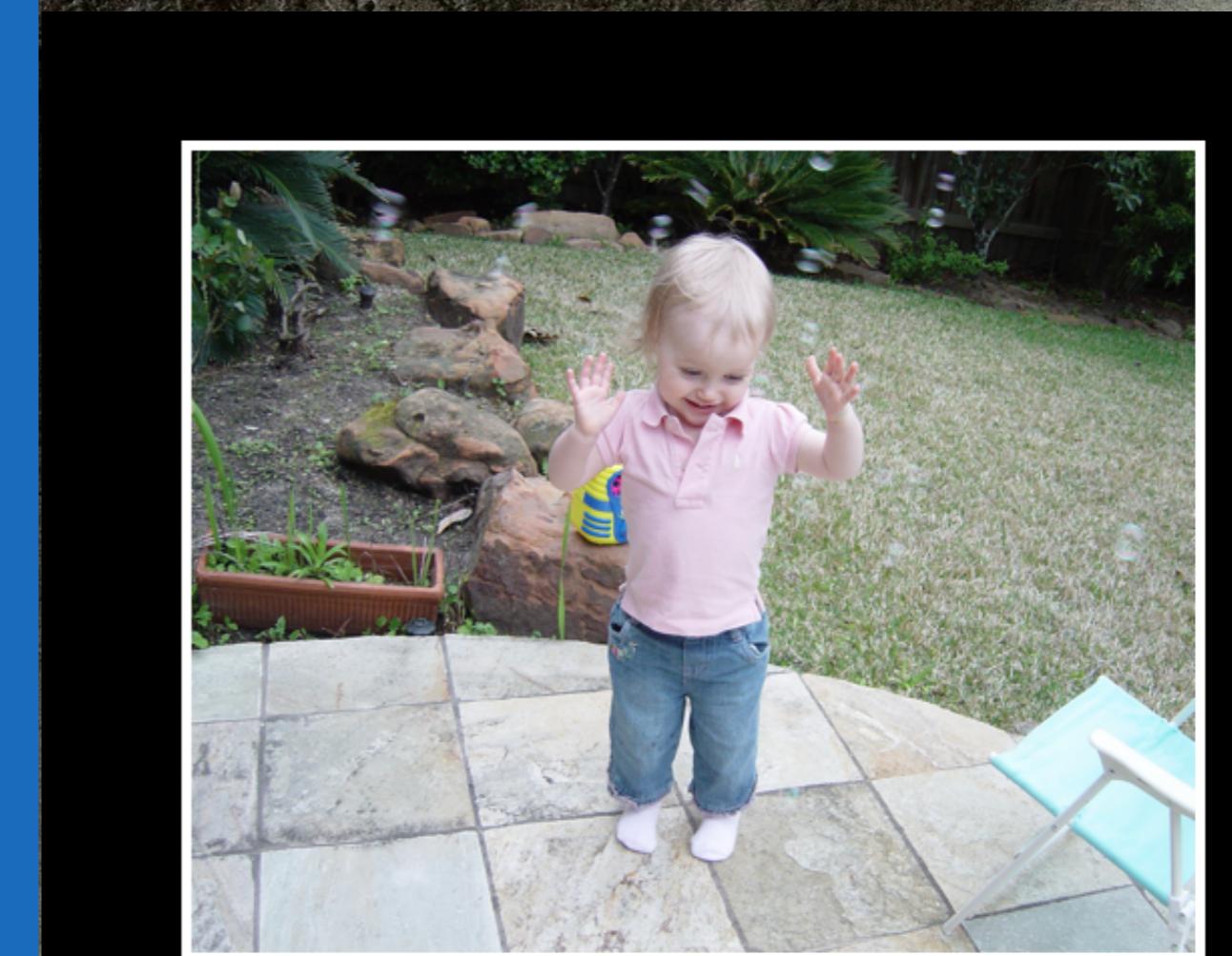
```
text-shadow:  
 10px /* X Versatz */  
 10px /* Y Versatz */  
 10px /* Blurradius*/  
 rgba(64, 64, 64, 0.5)  
;  
  
box-shadow:  
 10px /* X Versatz */  
 10px /* Y Versatz */  
 10px /* Blurradius*/  
 rgba(0, 0, 128, 0.25);
```



# REFLEKTION

```
box-reflect:  
below | above | left | right  
5px;  
  

```



# FILTER FUNKTIONEN

```
filter : blur(5px)
         brightness(0.2)
         saturate(50%)
         hue-rotate(30deg)
         contrast(125%)
         invert(100%)
         greyscale(100%)
         sepia(100%)

;

// Ggf. müssen Browser-Prefixes verwendet werden:
// -moz-filter, -webkit-filter, -o-filter, -ms-filter
```

# BACKGROUNDMISCHMODUS

```
.blended {  
background-image : url(face.jpg);  
background-color : red;  
background-blend-mode : multiply;  
}  
  
// Weitere Modus sind : normal , multiply, screen, overlay,  
// darken, lighten, color-dodge, saturation, color, luminosity
```

# ELEMENTMISCHMODUS

```
// Für Elemente gilt die Eigenschaft  
mix-blend-mode : normal;
```

Die Modus sind ähnlich: multiply, screen, overlay, darken, lighten, color-dodge ...

<https://css-tricks.com/basics-css-blend-modes/>

# ANIMATIONEN UND ÜBERGÄNGE

# CSS TRANSITION ANIMATIONEN UND EFFEKTE

- Eine CSS Transition ändert den Wert einer CSS-Eigenschaft über eine festgelegte Zeit.
- Dabei entstehen Animationen wie das Ein- und Ausblenden und Bewegen von HTML-Elementen ohne Javascript oder Flash.



# AUFBAU EINER TRANSITION

```
#box.left { position: absolute; left: 0;      }
#box.right { position: absolute; left: 1000px; }

#box {
    transition : left 5s ease-in-out;
}

document.getElementById('box').className = 'left';
document.getElementById('box').className = 'right';
```

# TRANSITION PROPERTY, DURATION, TIMING

## **transition-property**

Die Eigenschaft, die während des Übergangs geändert wird.

## **transition-duration**

Die Zeit, in der ein Übergang stattfinden soll.

## **transition-delay**

Zeit bis zum Start des Übergangs.

## **transition-timing-function**

Zeitlicher Verlauf des Übergangs.





# EIN AUFKLAPPENDES MENU

```
.content-nav {  
    display : block;  
    height  : 0;  
    overflow: hidden;  
  
    transition-property : height;  
    transition-duration : 0.25s;  
    transition-timing-function : ease-in-out;  
}  
.menu-show {  
    display : block;  
    height  : 5em;  
}  
  
$('button#menu-toggle').on('click', function () {  
    $('.content-nav').toggleClass('menu-show');  
});
```





# ÜBERGÄNGE MIT STATES

```
<div class="button"><a href="#">Mit Transition</a></div>  
...  
.button a {  
    background-color: yellow;  
    transition-property: background-color;  
    transition-duration: 1s; }  
  
.button a:hover { background-color: red; }
```

# DIE EASING EFFEKTE

## **ease**

Ohne weitere Angaben starten Transitions mit CSS langsam, beschleunigen in der Mitte und werden am Ende wieder langsamer.

## **linear**

ist eine gleichförmiger Übergang,

## **ease-in**

beginnt langsam und wird schneller,

## **ease-out**

beginnt schnell und läuft langsamer aus

## **ease-in-out**

kombiniert das langsame Anlaufen mit einem langsamen Auslaufen, aber einem deutlich schnelleren Teil in der Mitte.

## **cubic-bezier(x1,y1,x2,y2)**

ist Fine-Tuning für den Verlauf der Animation

# ANIMATIONEN

```
@keyframes my-animation {  
    0% { background-color:black; }  
    25% { background-color:red; }  
    50% { background-color:green; }  
    75% { background-color:yellow; }  
    100% { background-color:black; }  
}  
  
.background {  
    animation: my-animation 4s infinite;  
}
```

# TRANSFORMATIONEN

```
transform : rotateY(45deg);  
transform : scaleX(2.5%);  
transform : translate3d(0, 0, 90px);  
transform : perspective(500px)  
  
#threed-example {  
    transform : rotateZ(5deg);  
    transition : transform 2s ease-in-out;  
}  
  
#threed-example:hover {  
    transform : rotateZ(-5deg);  
}
```

<https://davidwalsh.name/demo/css-cube.php>

– DAVID WALSH

das war's mit CSS ...