

SASS UND CSS ARCHITEKTUR

SELEKTOREN

CSS SELEKTOREN

body {}	→ Elementeselektor
.my-class {}	→ Klassenselektor
a[href]	→ Attributselektor (minified)
:hover	→ Pseudoklasse (Zustand)
#my-id {}	→ ID–Selektor

input[type=email] → Attributselektor
[data-my-attr="value"] → Attributselektor

::before, ::after → Pseudoelemente

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

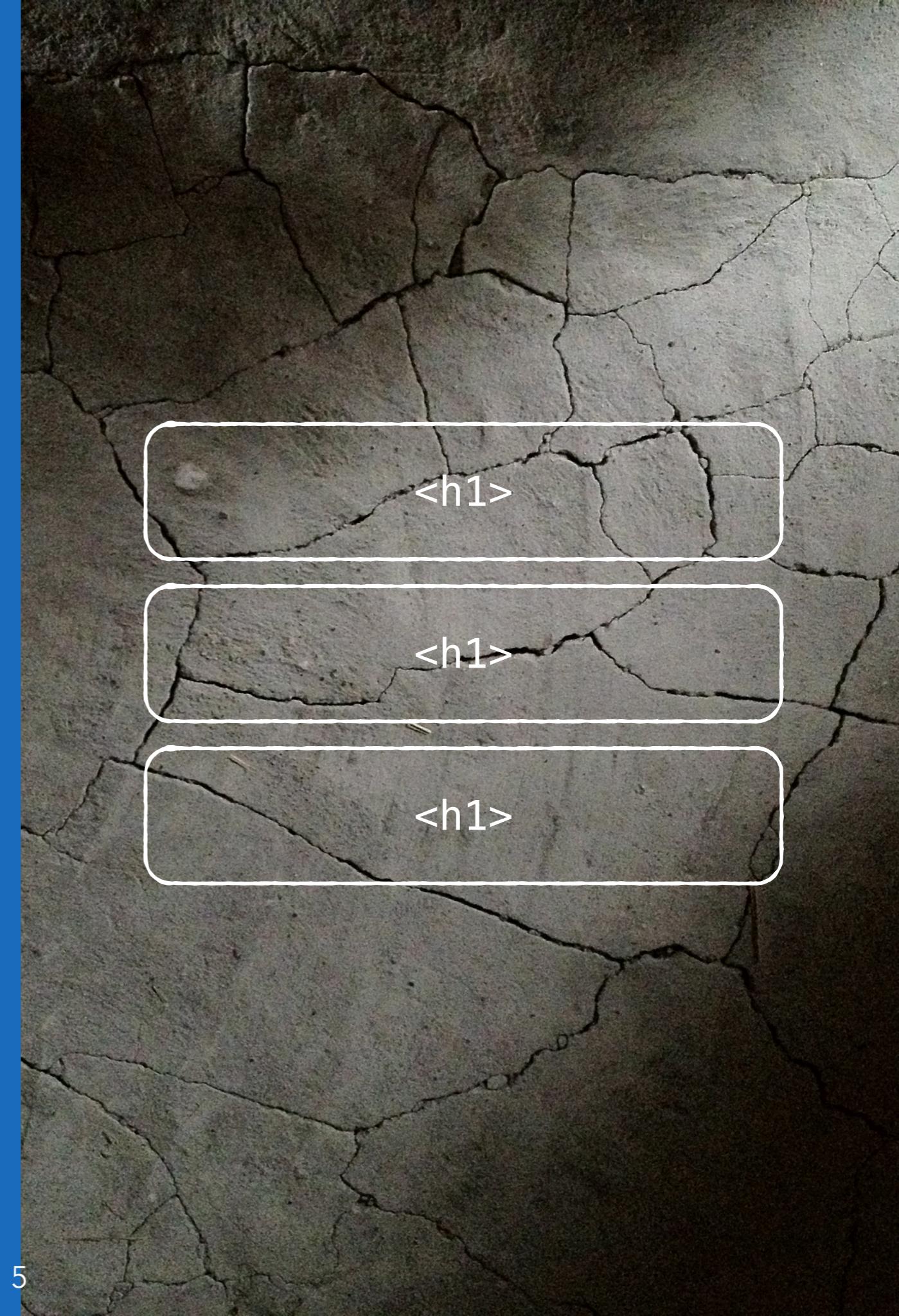
```
body * { ... }
```

ELEMENT SELEKTOR

body

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

-> h1 {}

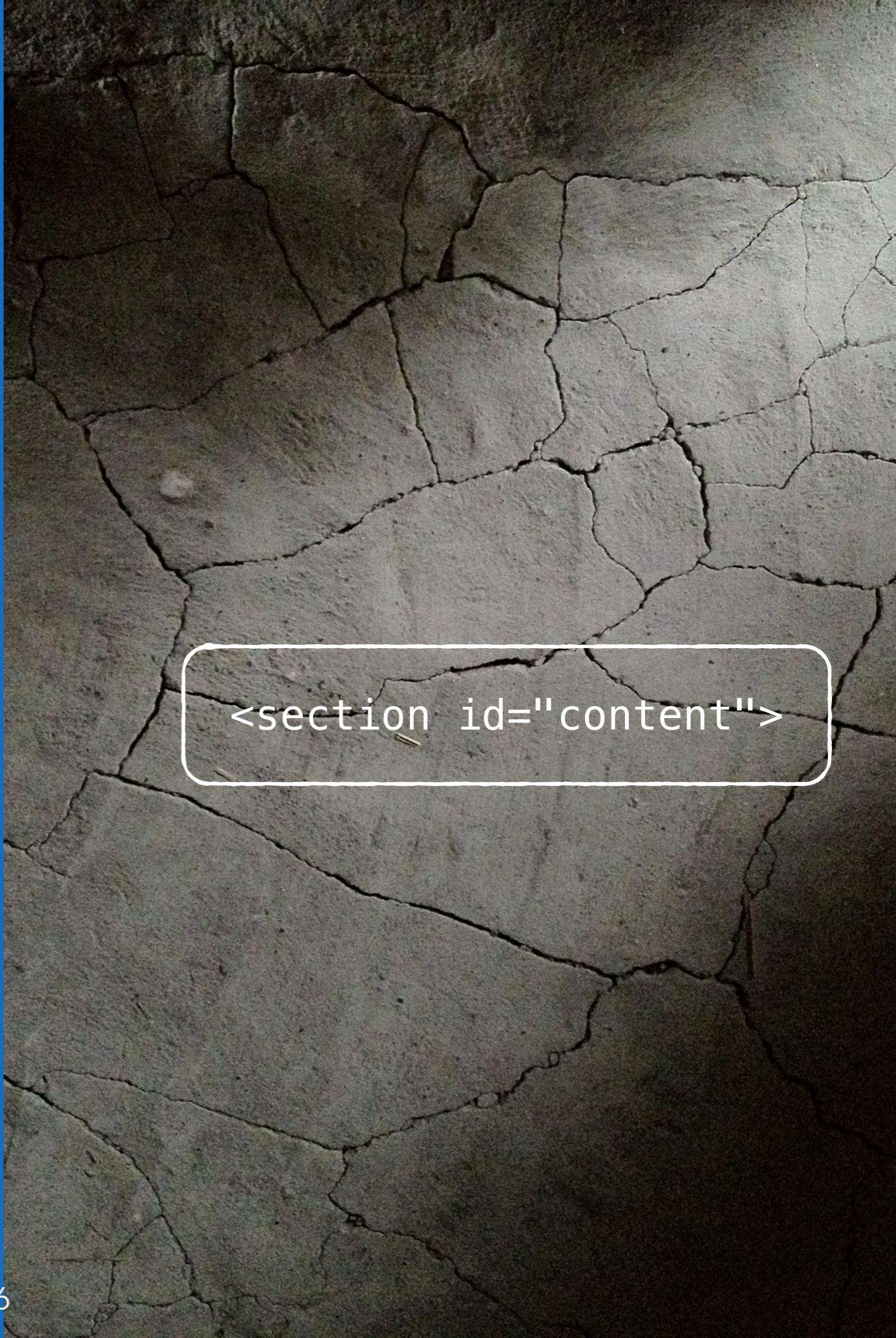


ID SELEKTOR

#my-id

Eine ID muss im Dokument unique sein!

→ section#content {}
→ #content {}



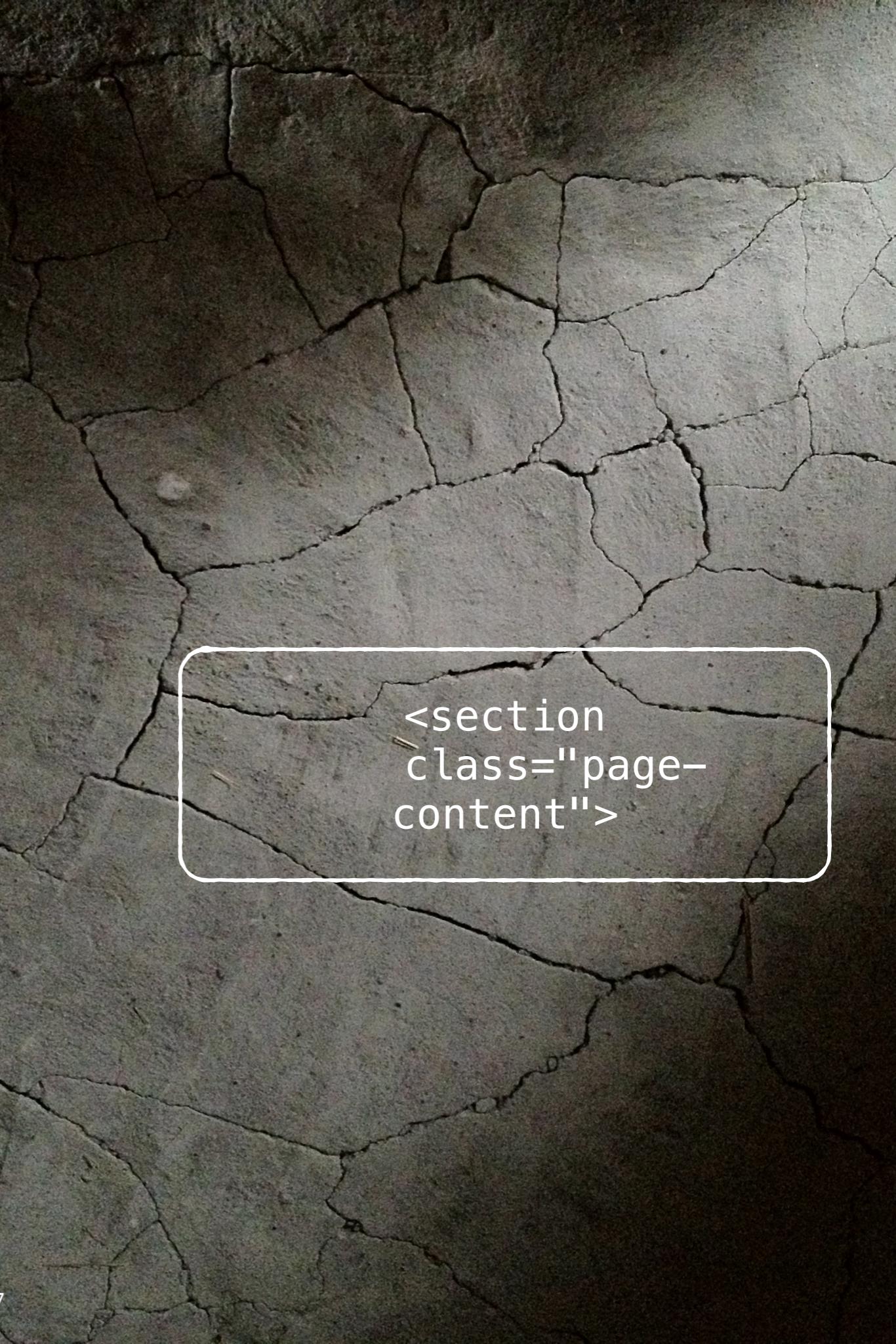
<section id="content">

KLASSENSEL KTOR

.my-class-name

Klassen sind mehrfach
verwendbar.

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



ATTRIBUTSELEKTOREN

[foo]

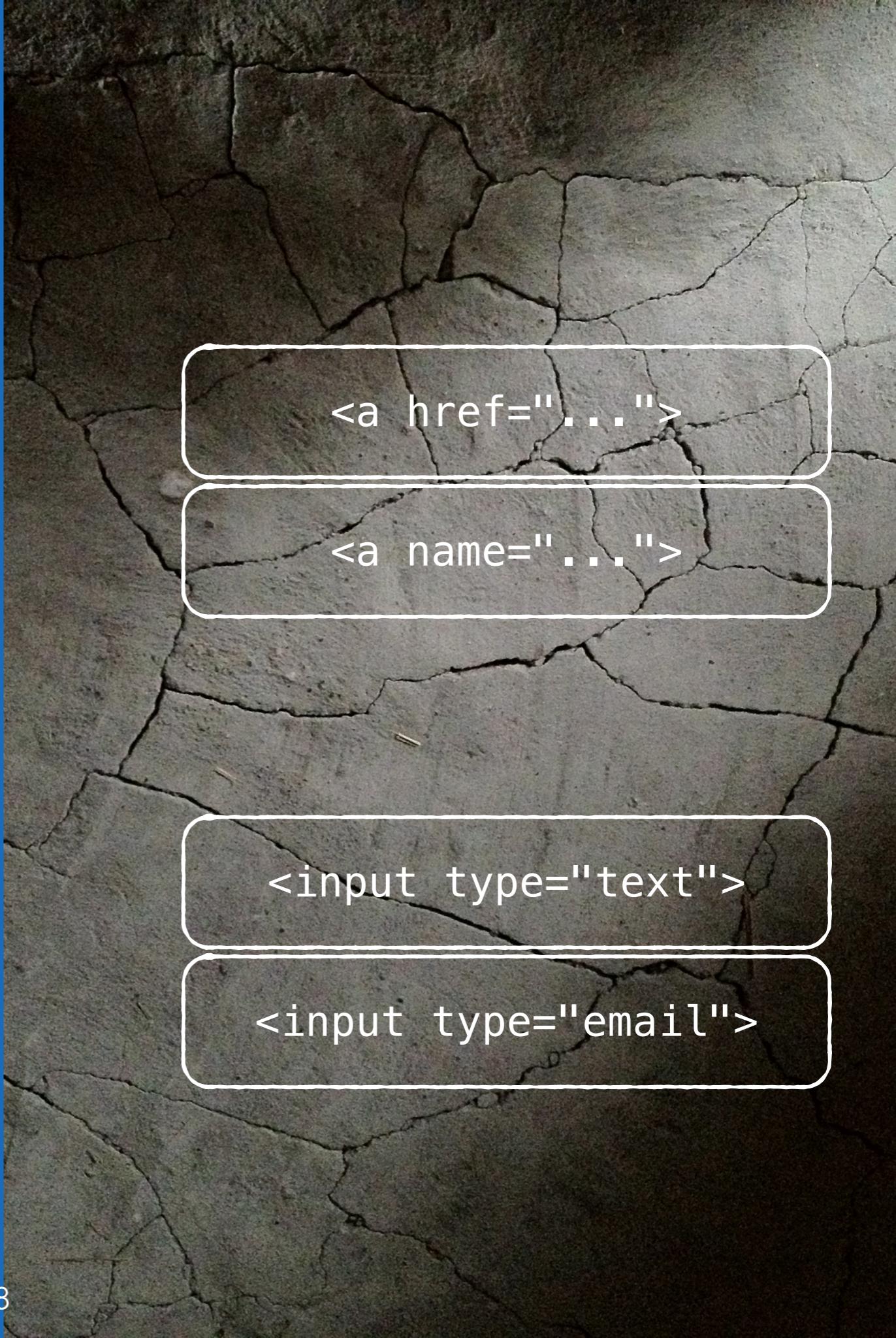
Stimmt mit jedem E-Element überein, dessen Attribut „foo“ gesetzt ist (ganz gleich, welchen Wert es hat).

- 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">
```

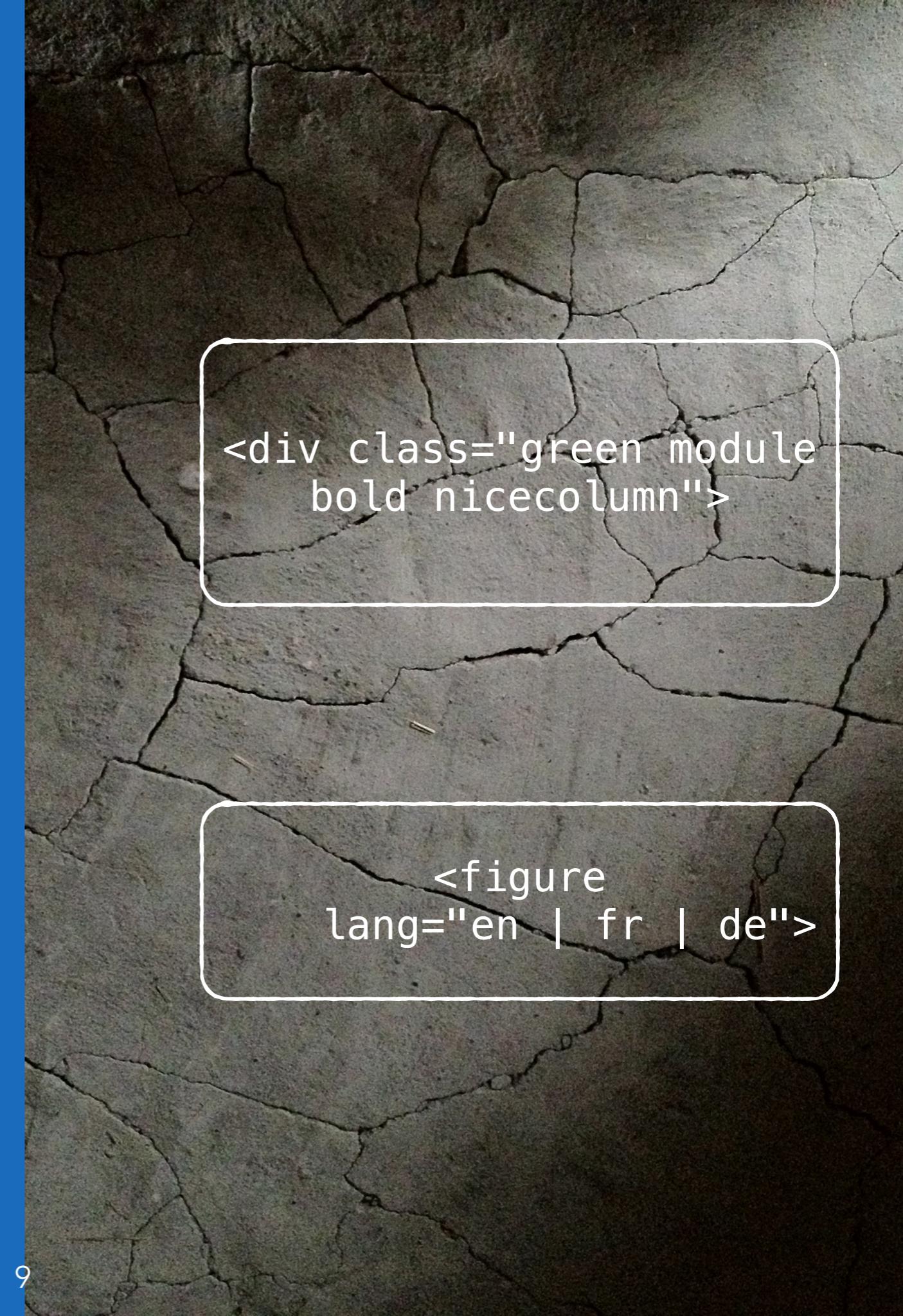
ATTRIBUTSELEKTOREN

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

`.group label` → Descendant

`.nav li` → Descendent

`ul 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

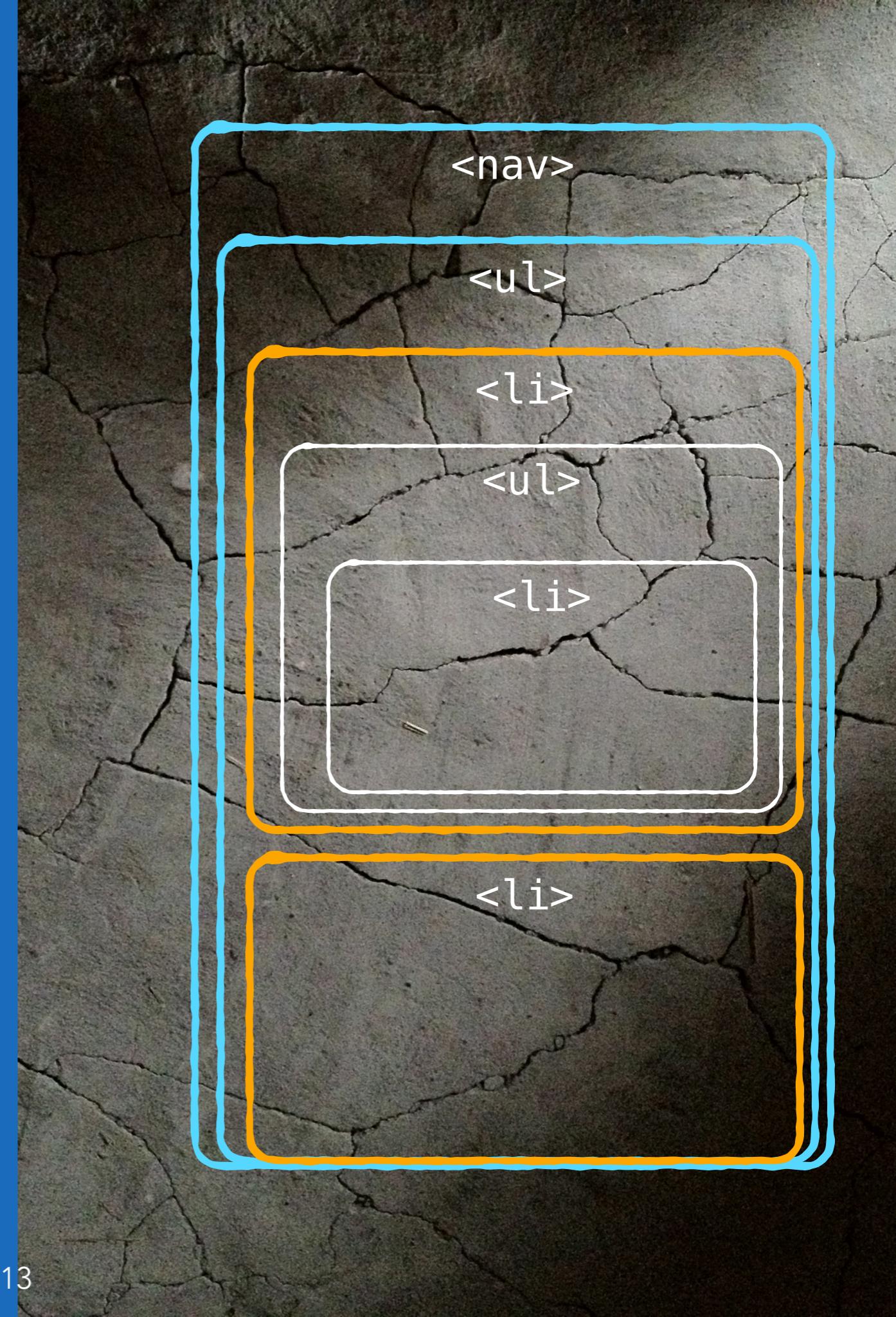
```
-> nav li { list-style :  
none; }
```



> CHILD SELECTOR

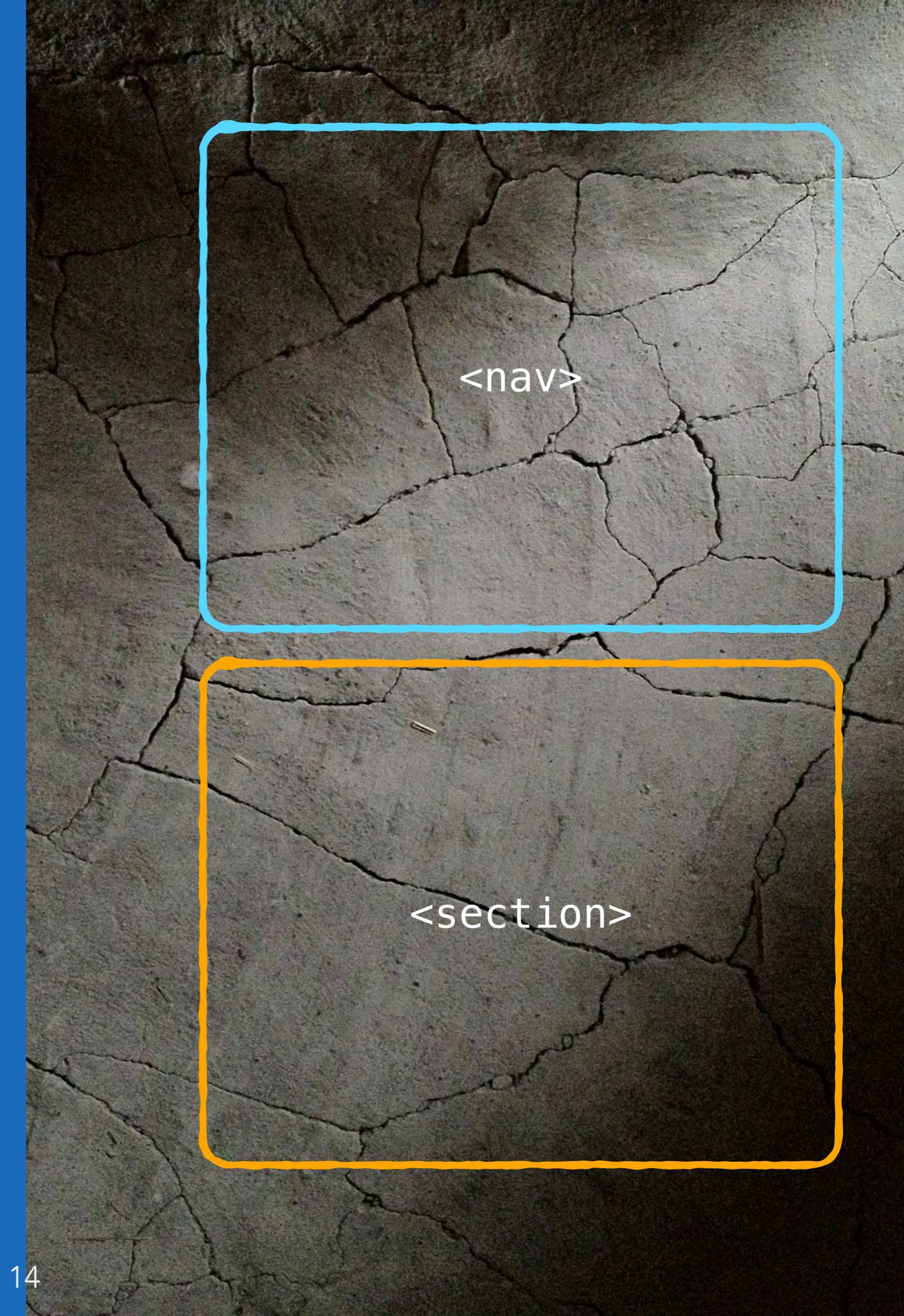
Stimmt mit allen F-Elementen überein, die Kindelemente eines Elements E sind.

→ nav > ul > li {}



+
ADJACENT SIBLING
(IMMEDIATE
FOLLOWER)

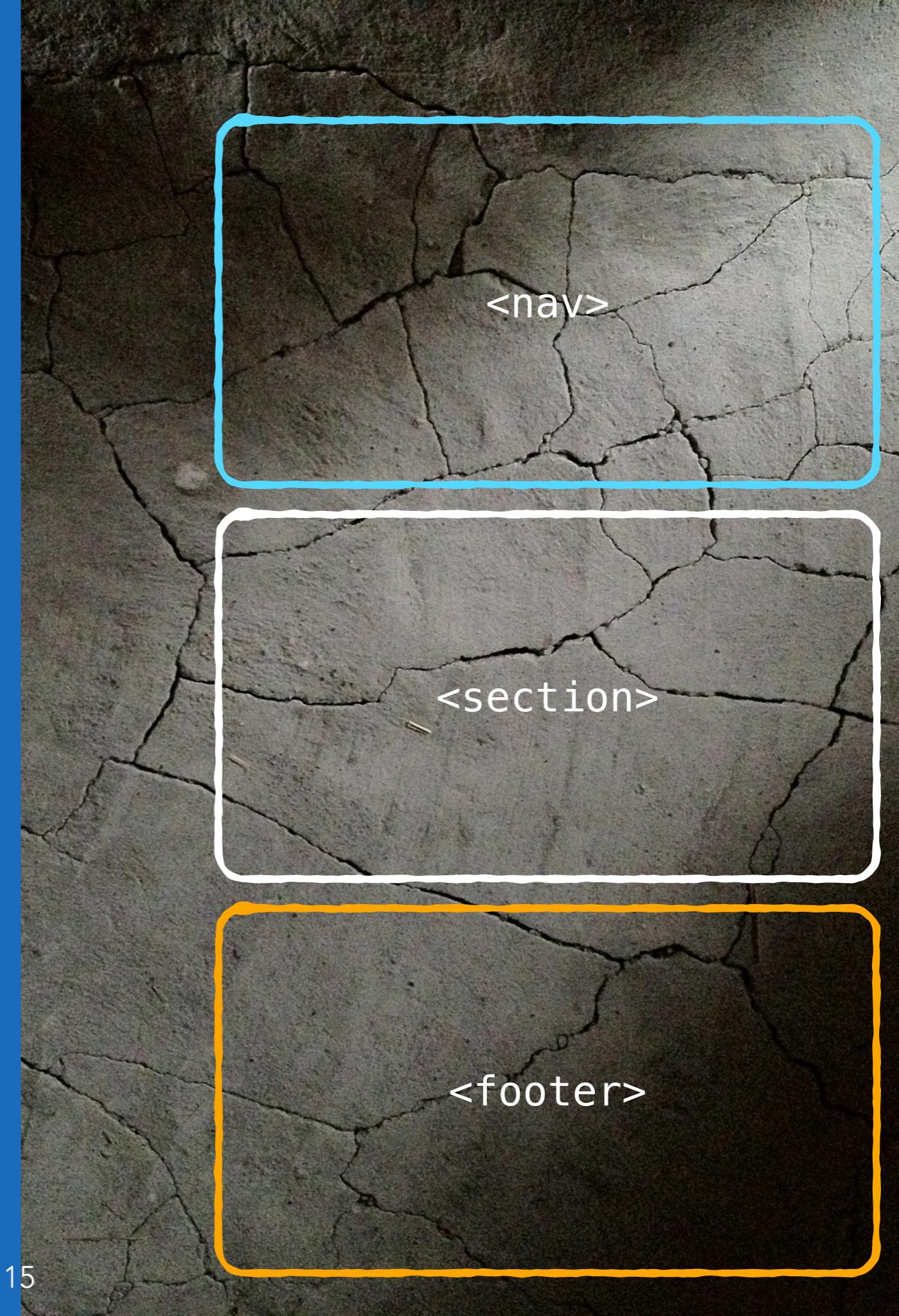
Stimmt mit jedem F-Element
überein, dem unmittelbar
ein Element E vorausgeht.
→ nav + section {}



~ 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
```

SELEKTIEREN WERDEN VON RECHTS NACH LINKS ABGEARBEITET

```
.nav.nav-row a[href] {}
```



```
1: ['href', 'href', 'href', 'href', 'href', 'href']
2: ['href', 'href', 'href', 'href', 'href', 'href']
3: ['href', 'href', 'href', 'href', 'href', 'href']
4: ['href', 'href', 'href', 'href', 'href', 'href']
```

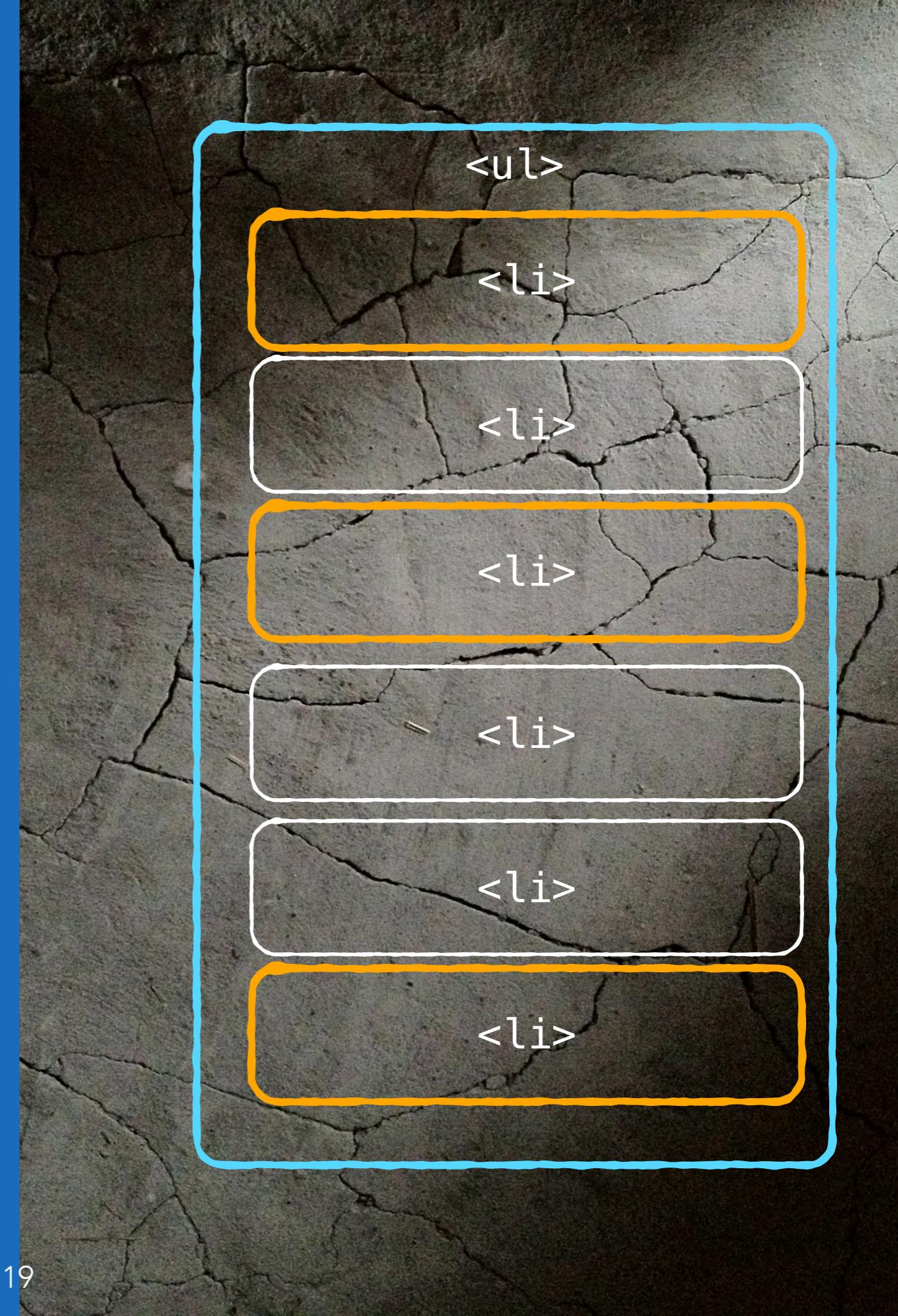
CSS SELEKTOREN PSEUDOKLASSEN

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

- Stateselector (Pseudoklasse)
- Pseudoelement
- Pseudoklasse
- Sprachattribut Selektor

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>
```

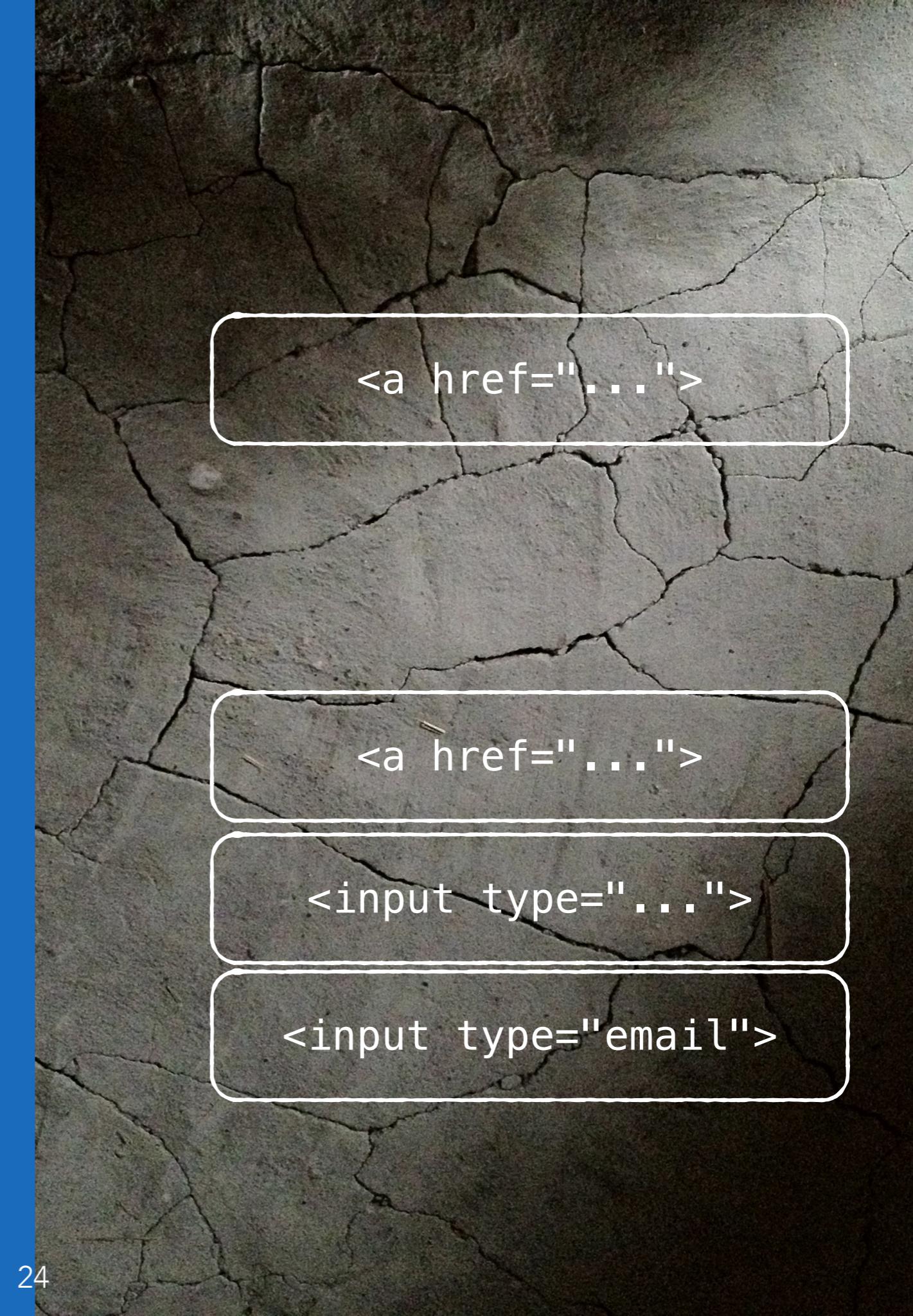
ZUSTANDS- SELEKTOREN

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.



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 (SPECIFICITY)

*	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
<p 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 */

Zustandsveränderungen 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.



Wie kann eine CSS für die Seitennavigation aufgebaut werden? Wie wird sie im HTML angewendet?

Entwickeln in einer navigation.css :

```
/* base */  
.nav { ... }  
  
/* layout */  
.nav-horizontal li { ... }  
.nav-vertical li { ... }  
  
/* states */  
.nav a[href] { ... }  
.nav a[href:visited] { ... }  
.nav a[href:active] { ... }  
.nav a[href:hover] { ... }
```

15 MINUTEN

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); }  
  
-webkit-border-radius: 10px;  
-moz-border-radius: 10px;  
-ms-border-radius: 10px;  
border-radius: 10px;
```

IMPORTS

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

// base.scss
@import 'reset';
@import 'variables';

body {
  font: 100% Helvetica, sans-serif;
  font-size: $font-size;
  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 cd path/to/web/directory  
  
$ npm install sass  
$ sass --watch assets/scss:assets/css --style=expanded  
  
--style=expanded | compact | compressed | nested  
--sourcemap=None
```

BLOCK - ELEMENT - MODIFIER NOMENKLATUREN FÜR CSS

BLOCK - MODIFIER

block--modifier

```
block
<button class="button">
  Normal button
</button>
<button class="button button--state-success">
  Success button
</button>
<button class="button button--state-danger">
  Danger button
</button>

<header> ... </header>
<form> ... </form>
```

class for block

```
.button {
  display: inline-block;
  border-radius: 3px;
  padding: 7px 12px;
  border: 1px solid #D5D5D5;
  background-image:
    linear-gradient(#EEE, #DDD);
  font: 700 13px/18px Helvetica, arial;
}

.button--state-success { --modifier
  color: #FFF;
  background: #569E3D
    linear-gradient(#79D858, #569E3D)
    repeat-x;
  border-color: #4A993E;
}

.button--state-danger {
  color: #900;
}
```

BLOCK - ELEMENT - MODIFIER

block_element--modifier

element

```
<form class="form form--theme-xmas form--simple">
  <input class="form__input" type="text" />
  <input
    class="form__submit form__submit--disabled"
    type="submit" />
</form>
```

```
.form { }
.form--theme-xmas { }
.form--simple { }
.form__input { }
.form__submit { }
.form__submit--disabled { }
```

PREFIX - BLOCK - ELEMENT - MODIFIER

prefix-block_element--modifier

prefix

```
<form class='dvz-form
    dvz-form--theme-xmas form--simple">
    <input class="dvz-form__input" type="text" />
    <input
        class="dvz-form__submit
            dvz-form__submit--disabled"
        type="submit" />
</form>
```

```
.dvz-form { }
.dvz-form--theme-xmas { }
.dvz-form--simple { }
.dvz-form__input { }
.dvz-form__submit { }
.dvz-form__submit--disabled { }
```

<http://getbem.com/>

[https://webuild.envato.com/blog/**how-to-scale-and-maintain-legacy-css-with-sass-and-smacss/**](https://webuild.envato.com/blog/how-to-scale-and-maintain-legacy-css-with-sass-and-smacss/)

RESPONSIVE LAYOUTS MOBILE FIRST APPROACH

MEDIEN ATTRIBUTE EINSETZEN

really very tiny devices:

@media screen and (max-width:480px) { ... } including the device

tiny devices:

@media screen and (min-width:481px) { ... }

small devices:

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

medium devices:

@media screen and (min-width:1024px) { ... }

large devices:

@media screen and (min-width:1280px) { ... }

xtra large:

@media screen and (min-width:1600px) { ... }

full hd device:

@media screen and (min-width:1920px) { ... }

4K monitor:

@media Screen and (min-width:3840px) { ... }

MOBILE FIRST PROGRESSIVE ENHANCEMENT

PRINTER

```
@media print {}  
@media print and (min-width:16cm) {}
```

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) { ... }
```

HD READY

```
@media screen and (min-width:1440px) { ... }
```

??

```
@media screen and (min-width:1600px) { ... }
```

FULL HD READY

```
@media screen and (min-width:1920px) { ... }
```

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) { ... }

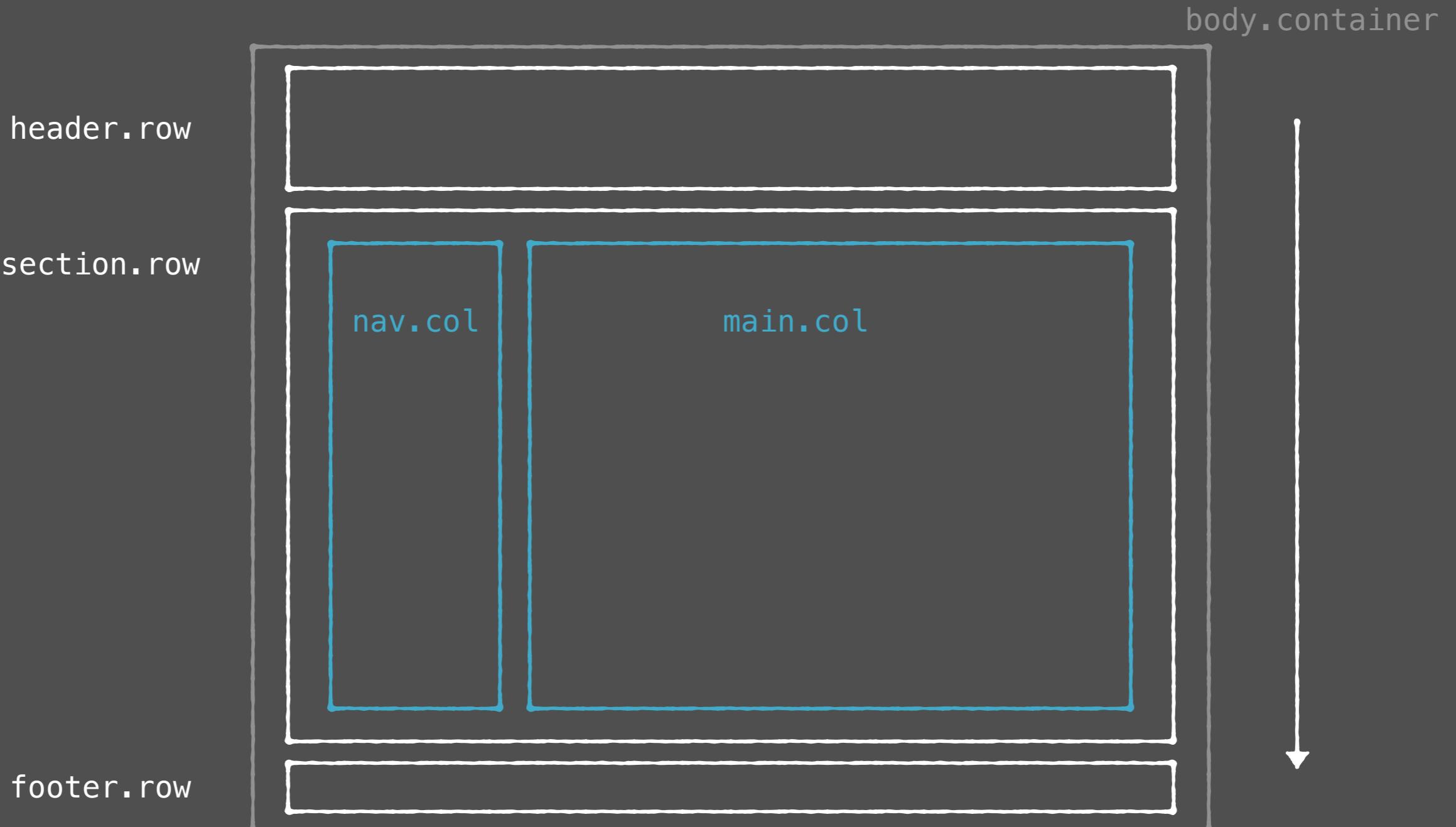
@media print and (resolution:300dpi) { ... }

@media screen and (min-resolution:96dpi) { ... }
@media screen and (min-resolution:192dpi) { ... } // Retina

@media print and (color) { ... }

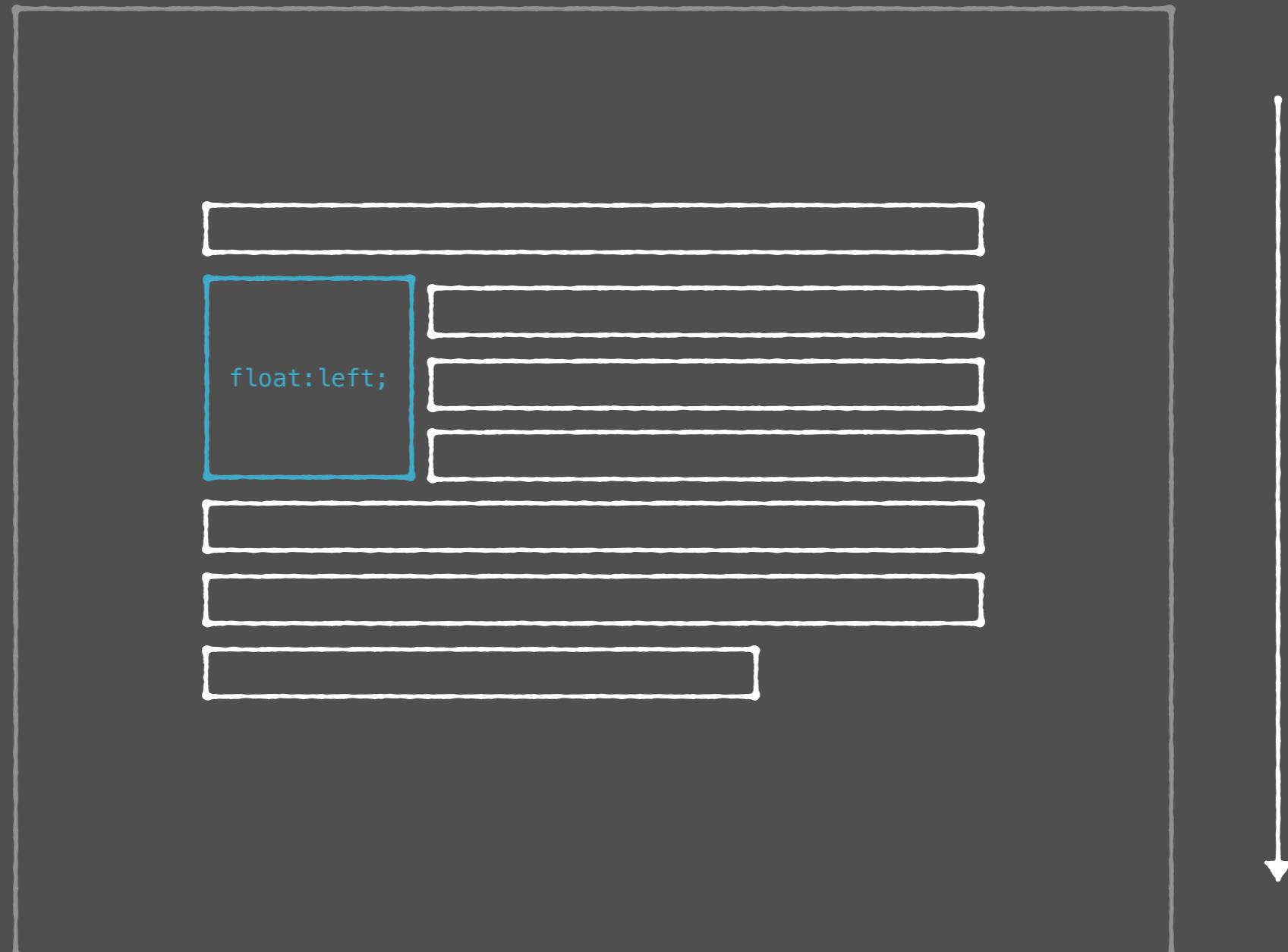
@media screen and (aspect-ratio:16/9) { ... }
```

Bootstrap Grid



p dolor sit lorem ipsum body.container

Float



Flexbox

```
container -> display: flex; flex-direction: column || row;
```



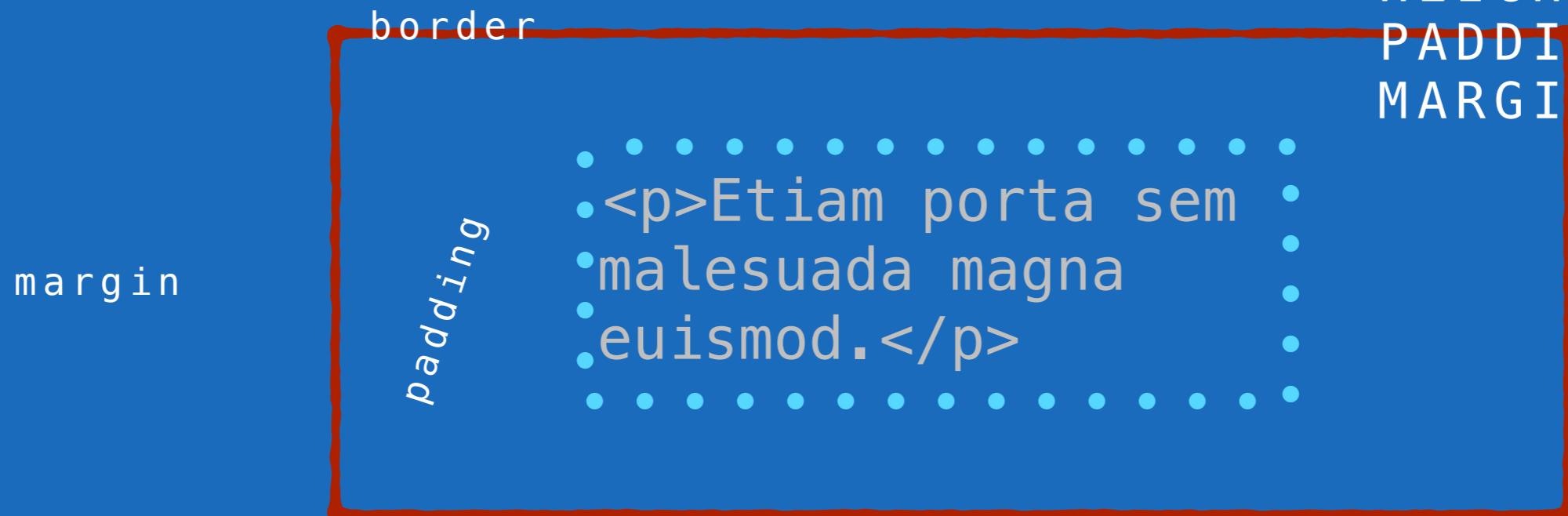


DISPLAY, WIDTH, HEIGHT, PADDING, MARGIN DAS BOX MODEL

DIE GRÖSSE VON HTML ELEMENTEN BERECHNEN

DAS BOX MODEL

```
box-sizing: content-box;  
box-sizing: border-box;
```



WIDTH	:	100PX
PADDING	:	2 X 2REM (1REM = 16PX)
BORDER	:	2 X 1PX

100PX!

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

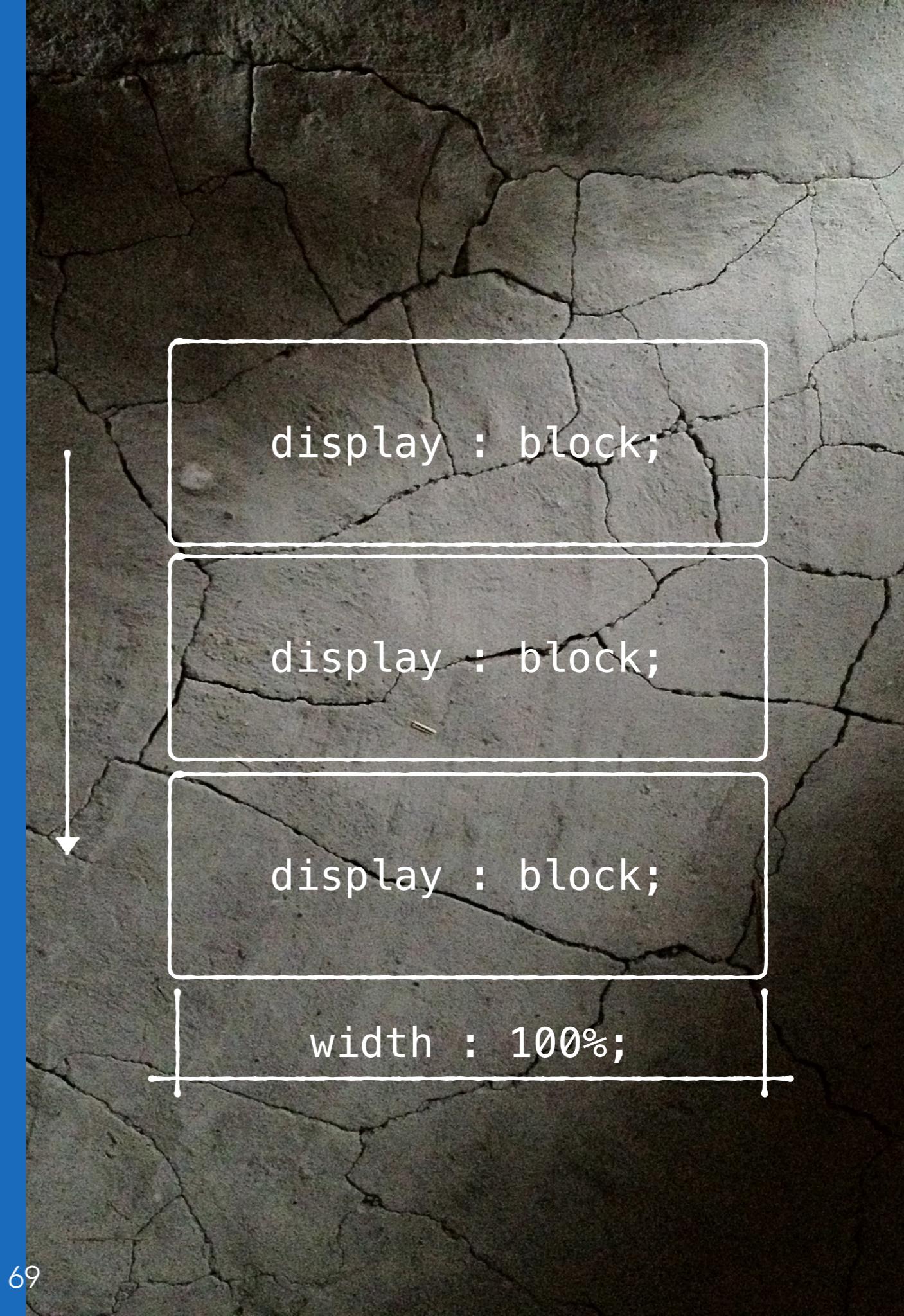
Padding mit in die width Dimension

DISPLAY, WIDTH, HEIGHT, PADDING, MARGIN

DISPLAY TYPEN

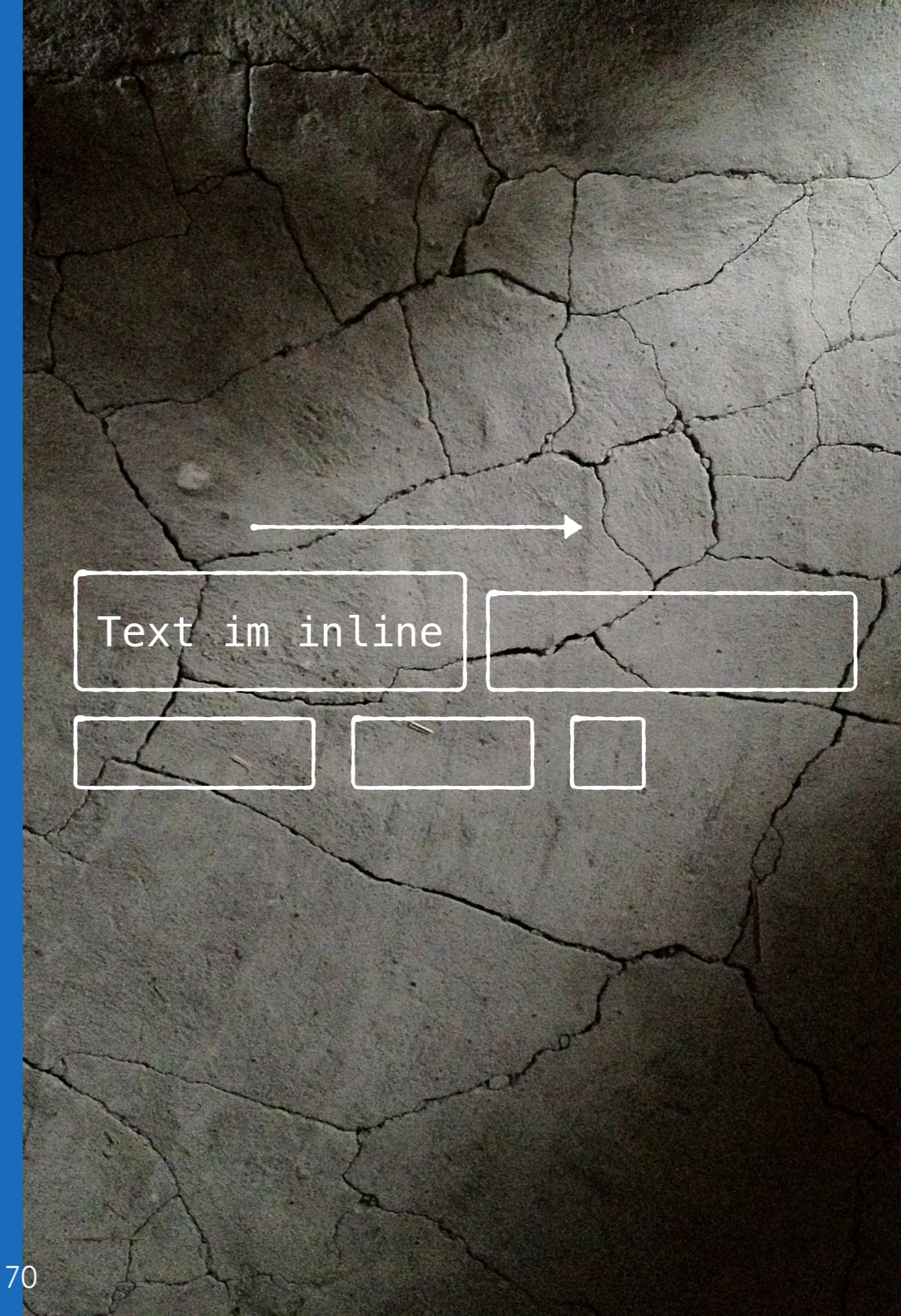
BLOCK

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



INLINE

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

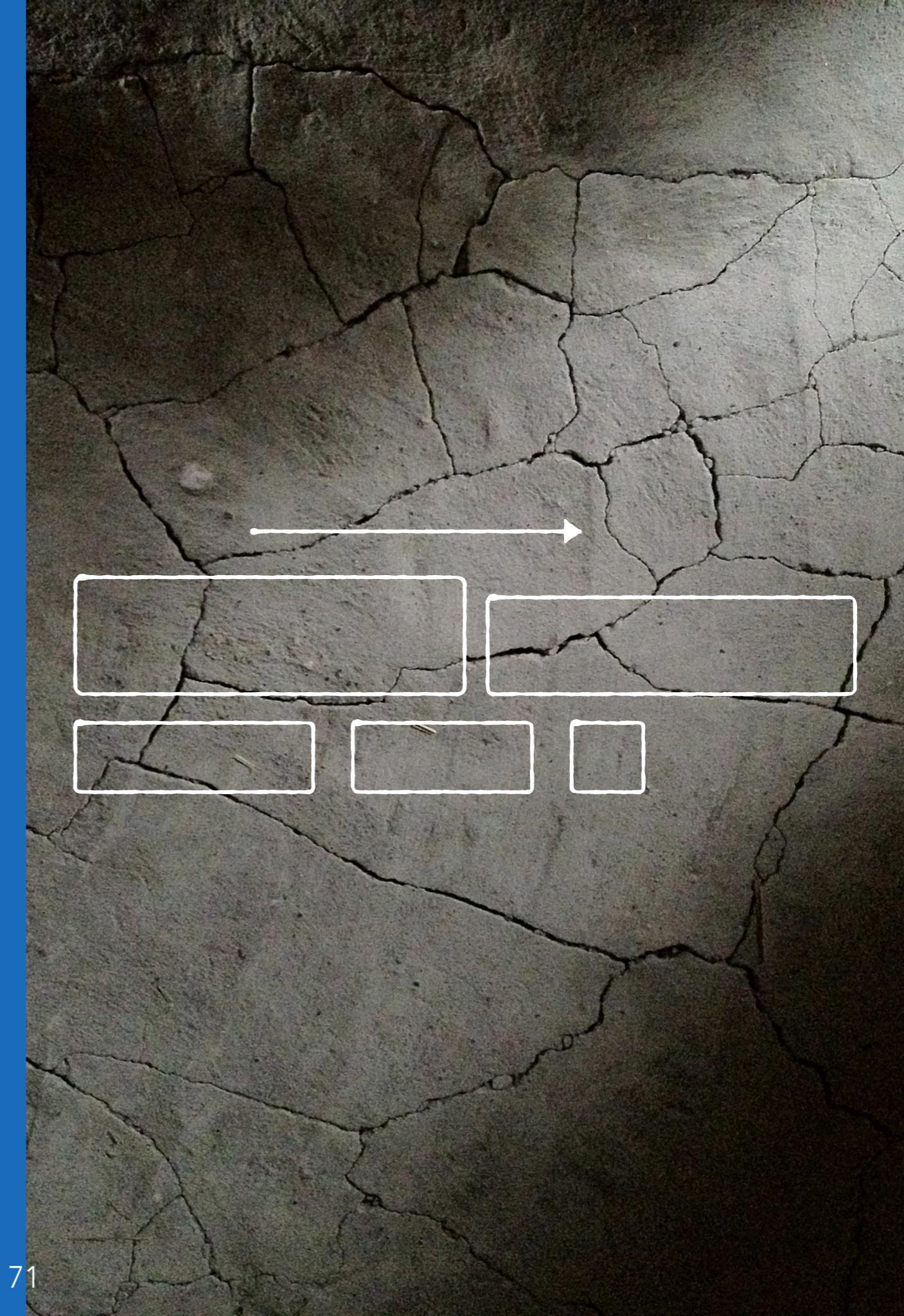


INLINE-BLOCK

```
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;  
}
```

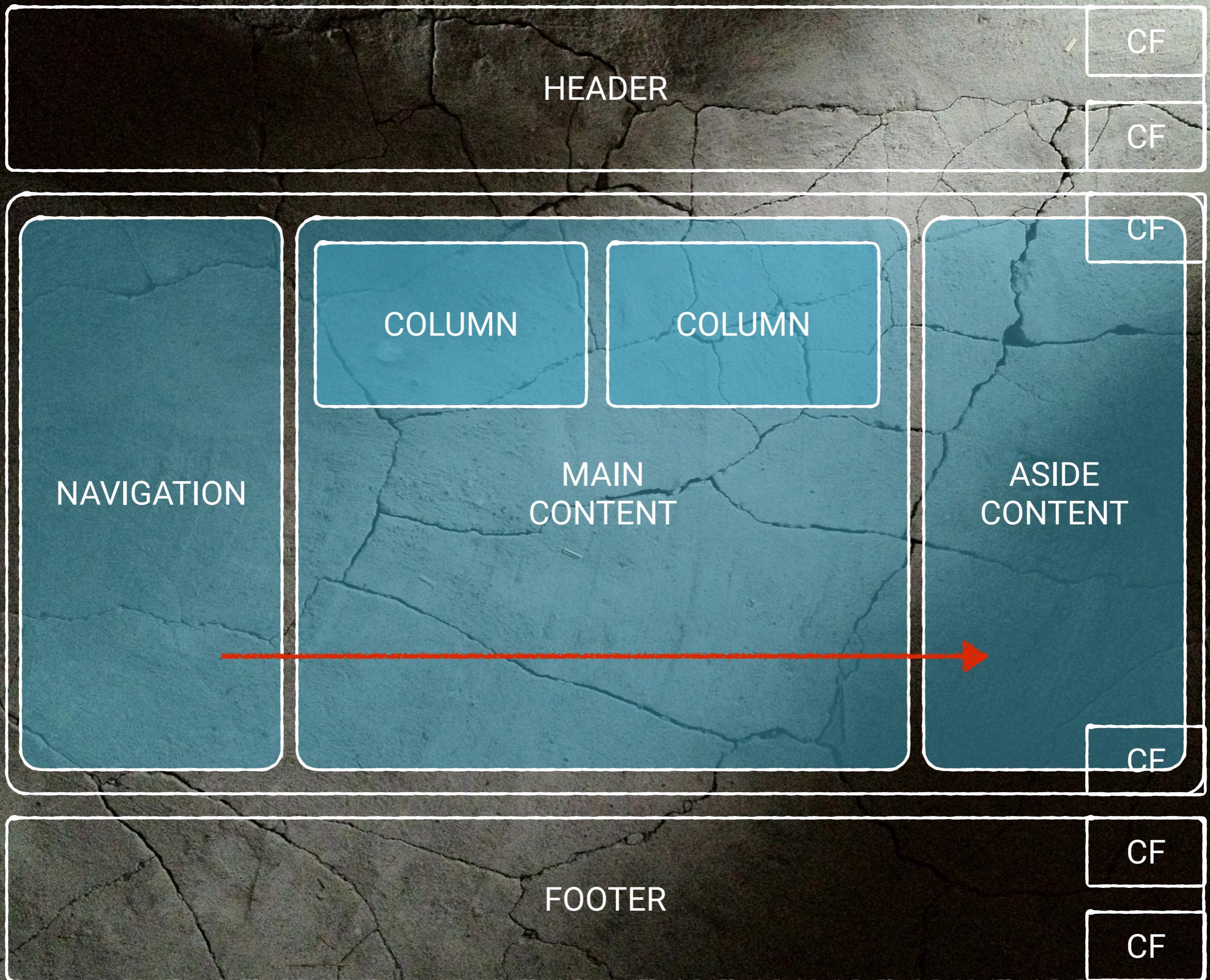
Ein Blockelement erhält eine Höhe und verdrängt damit Elemente, auch wenn nichts drin steht.
Lässt eine vertikale Zentrierung zu!

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

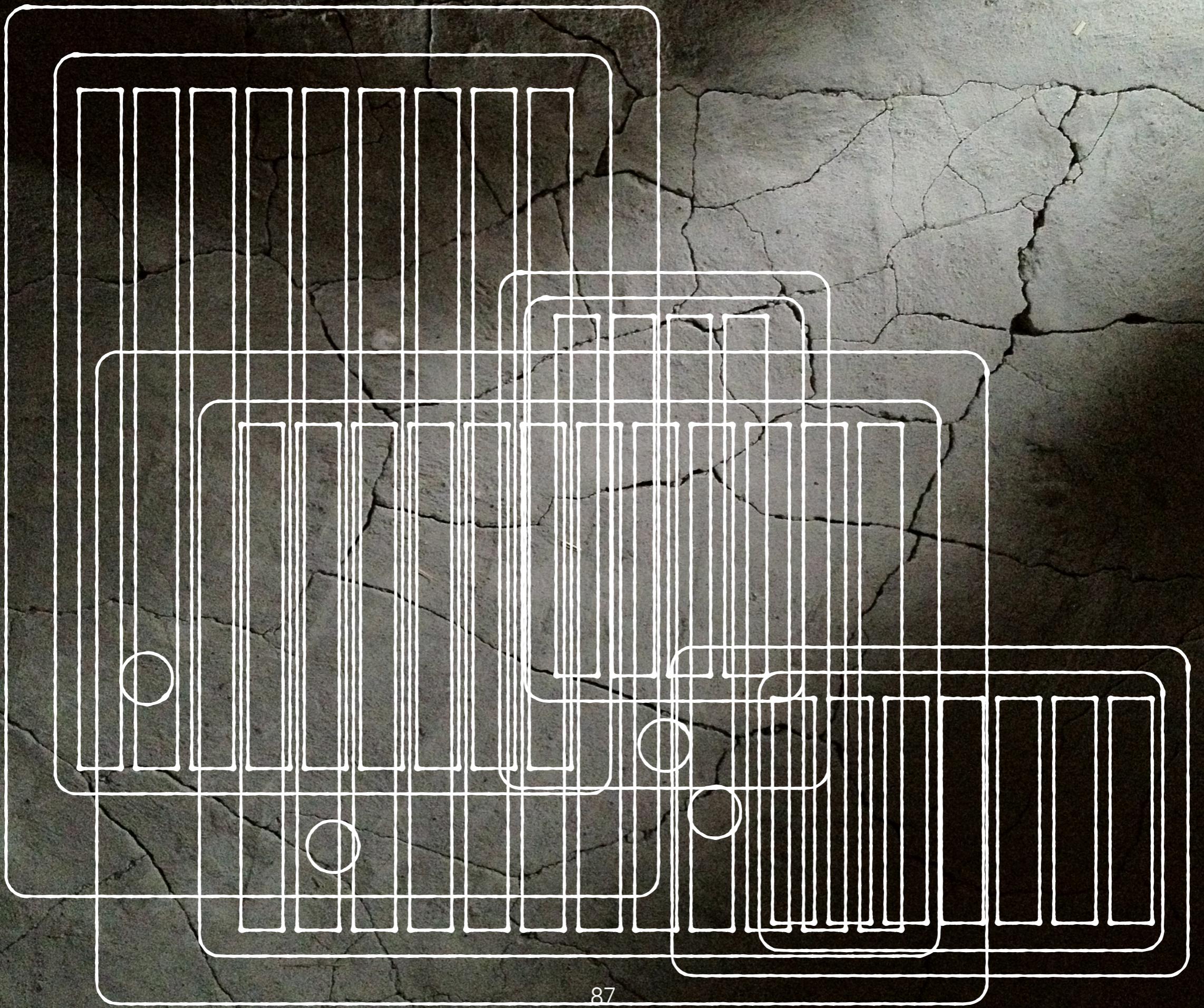
5 Spalten

5 Spalten

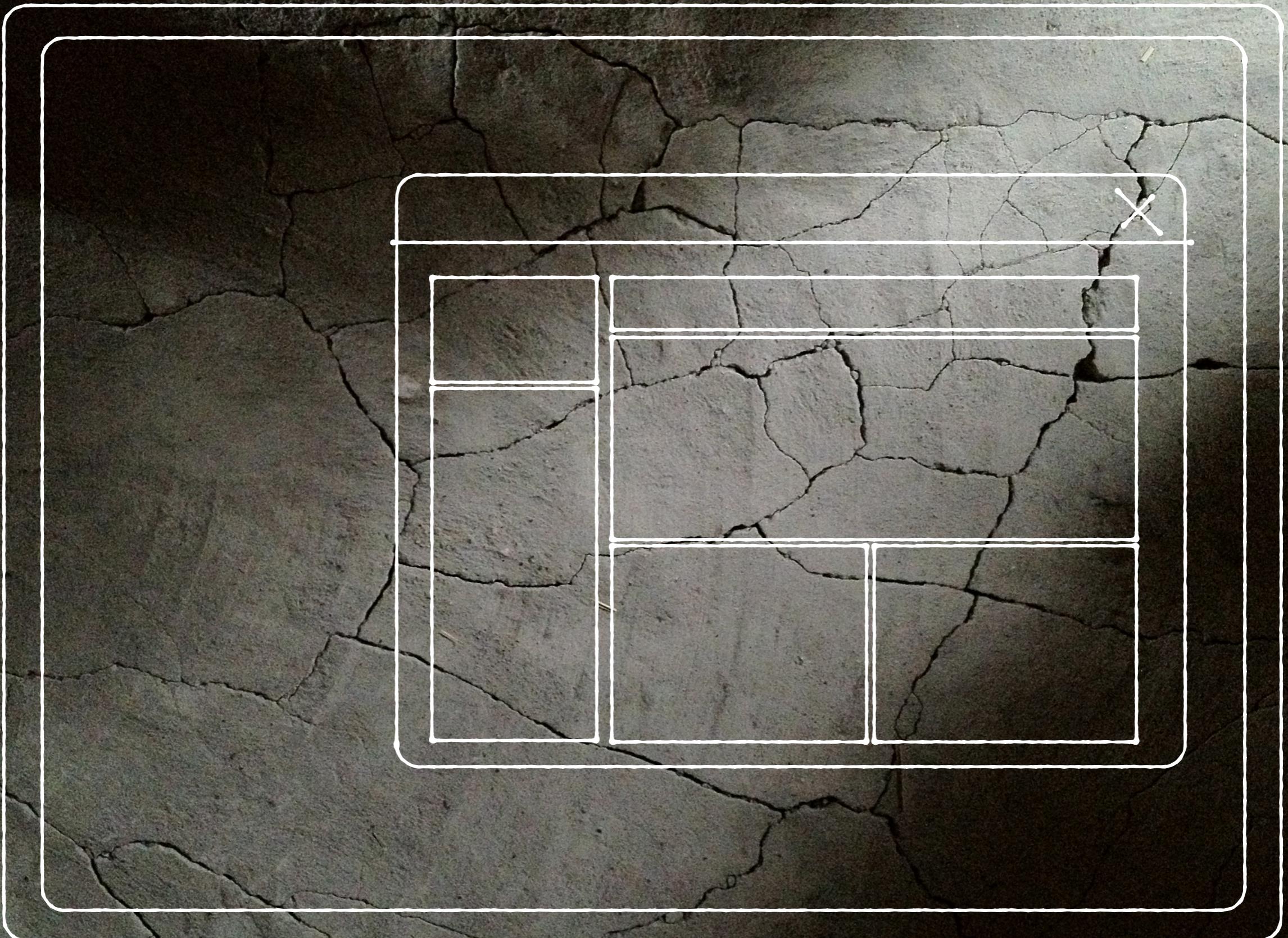
3 Spalten

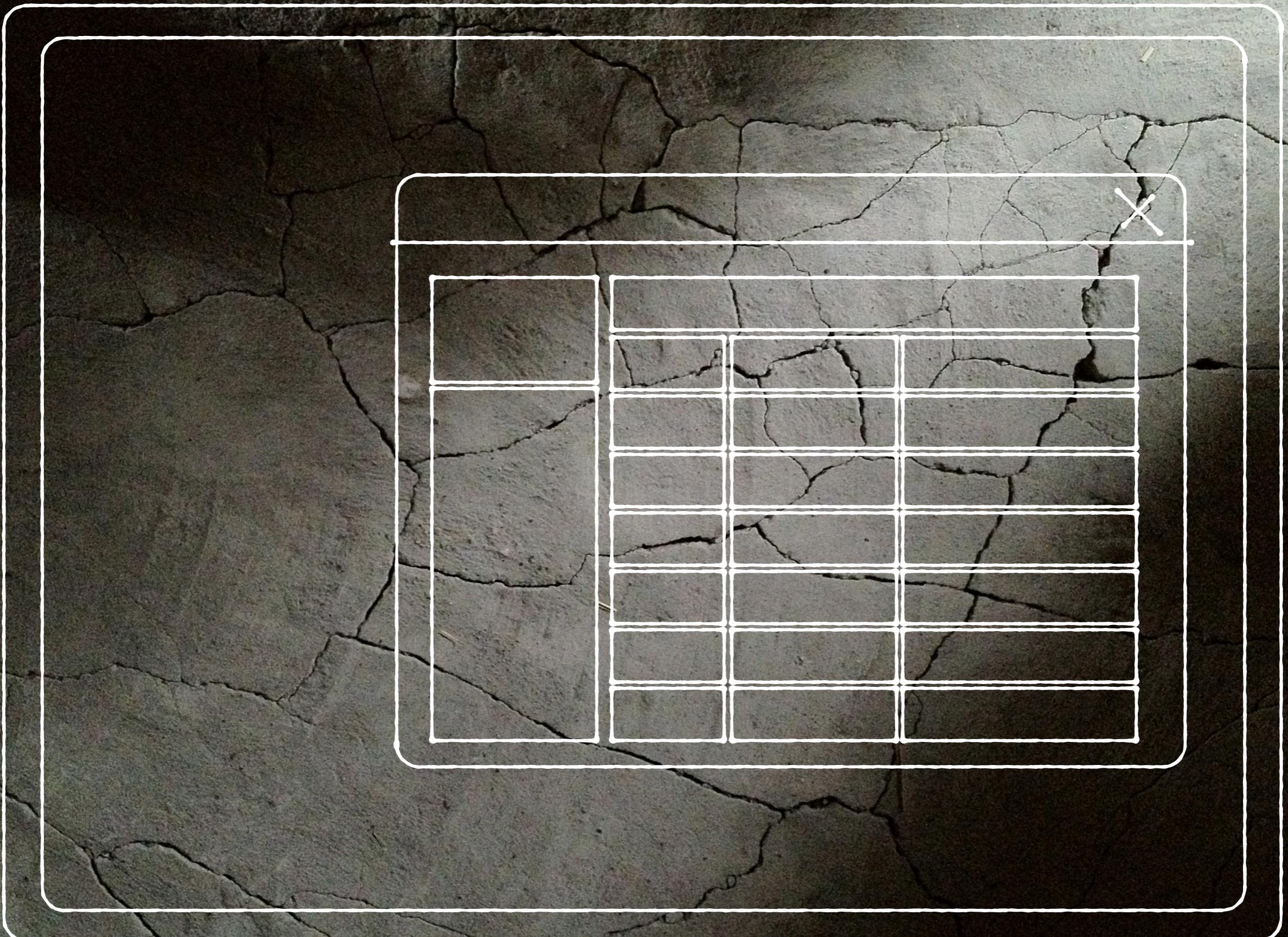












EIN GRID SCHREIBEN.

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

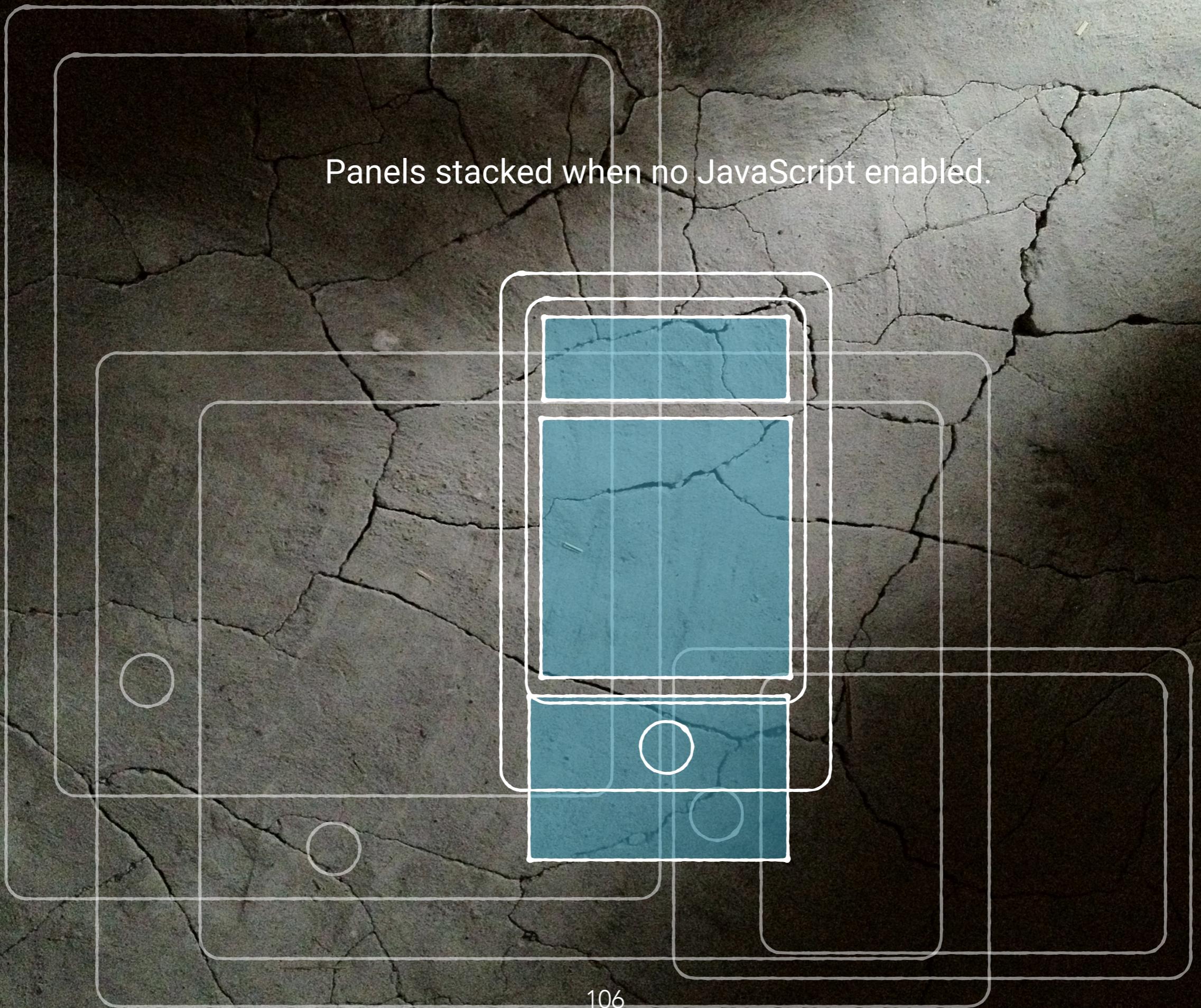
```
[class*="column"] + [class*="column"]:last-child {  
    float: right;  
}  
  
[class*="column"] + [class*="column"].end {  
    float: left;  
}
```

<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.

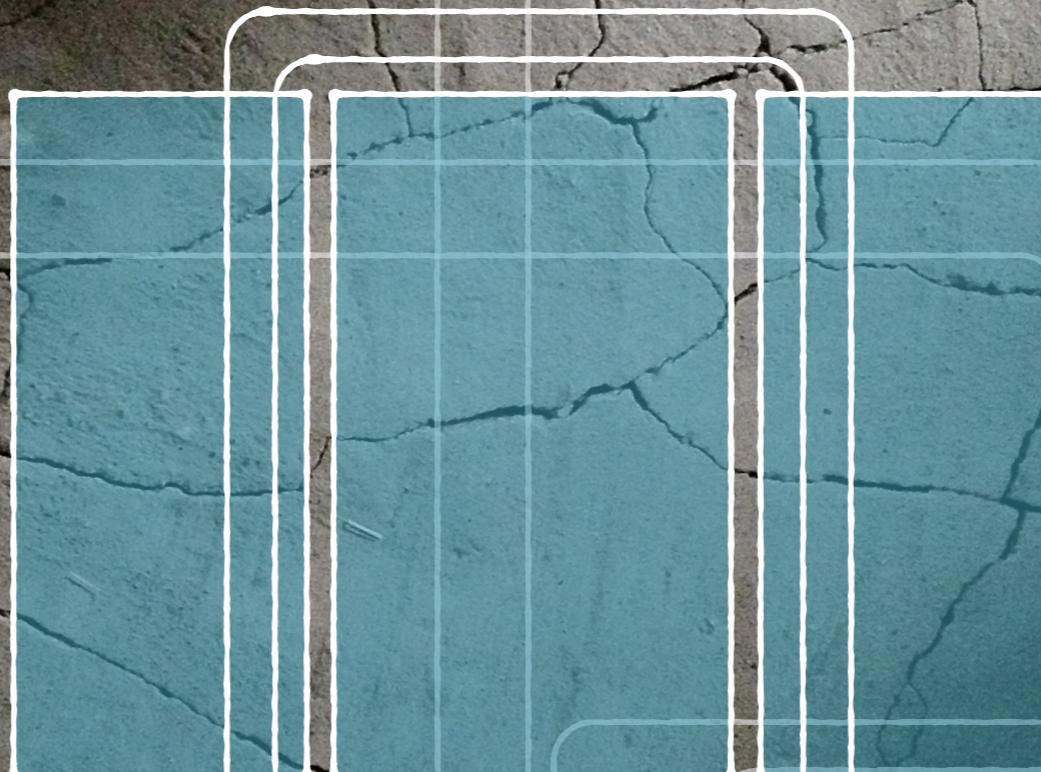


A RESPONSIBLE START

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.

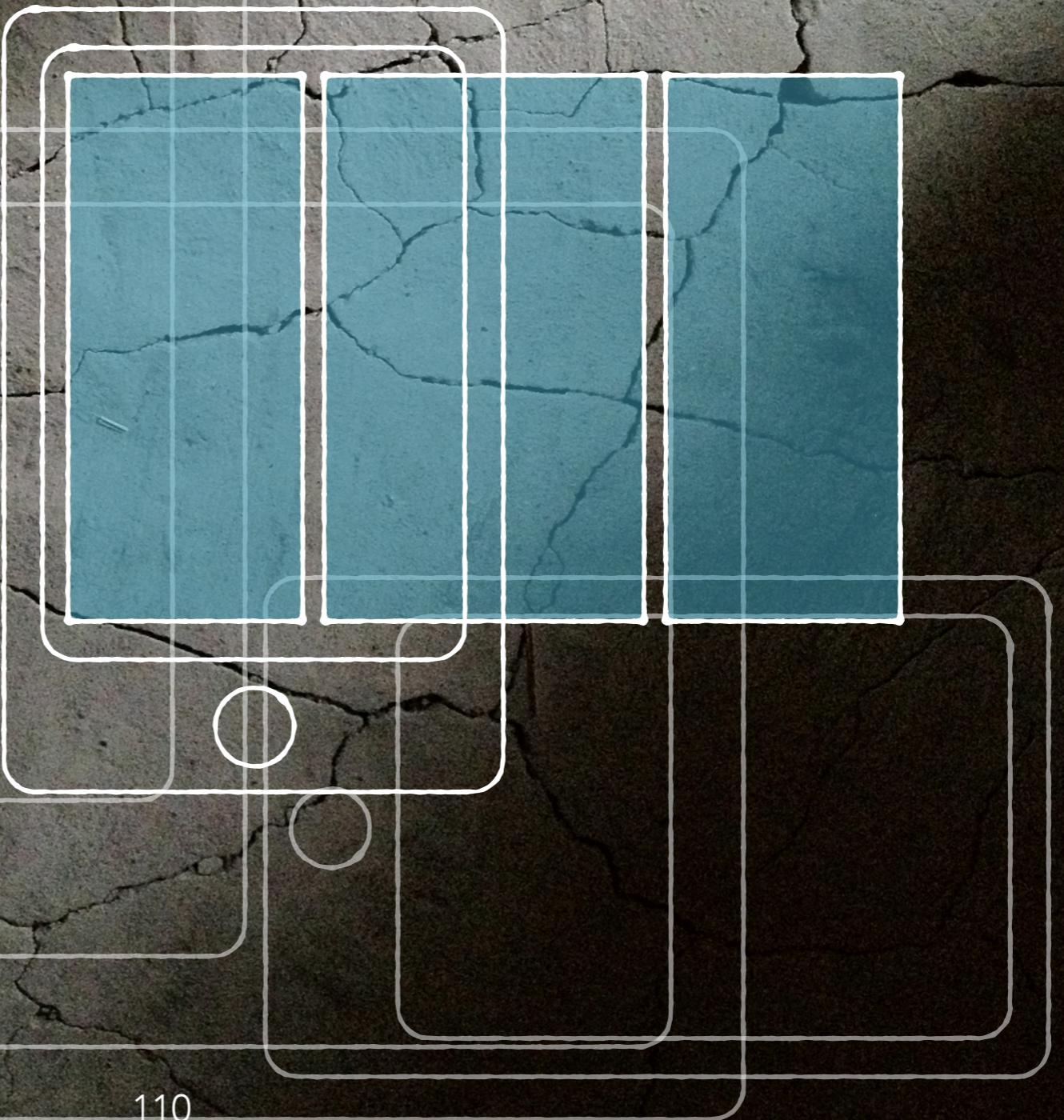


ACCESS LEFT

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.

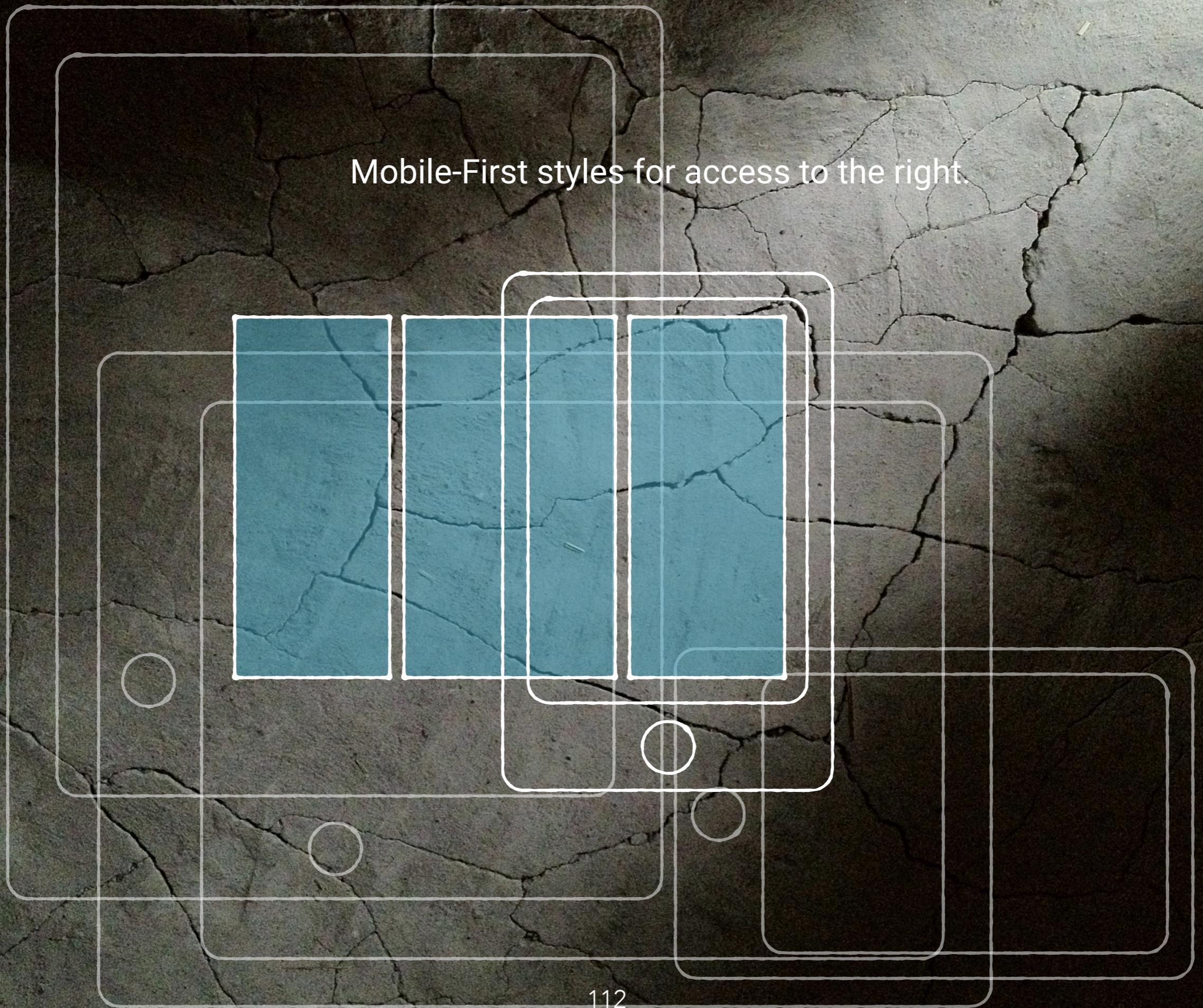


ACCESS RIGHT

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.

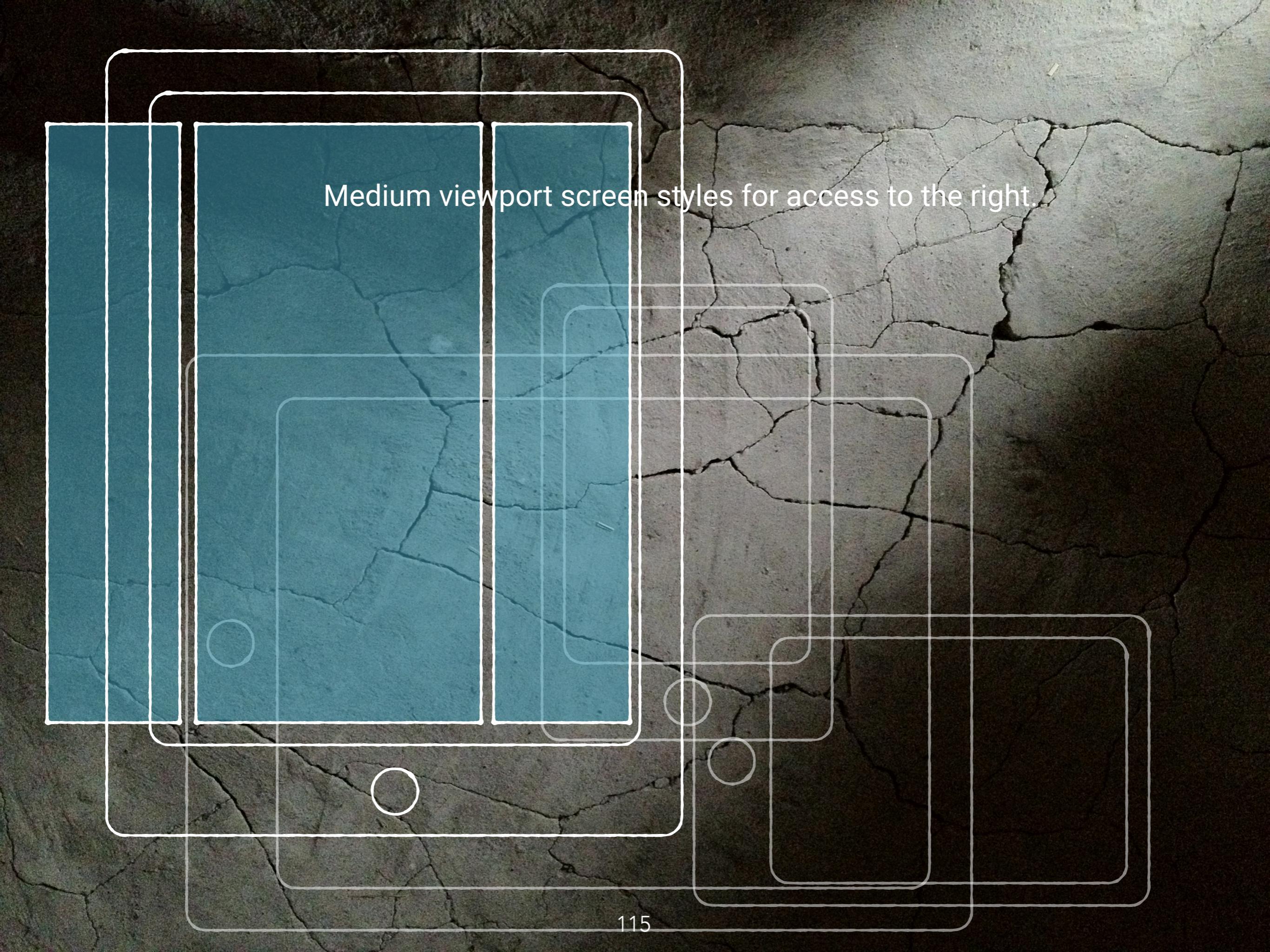


MEDIUM VIEWPORT STYLES

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.



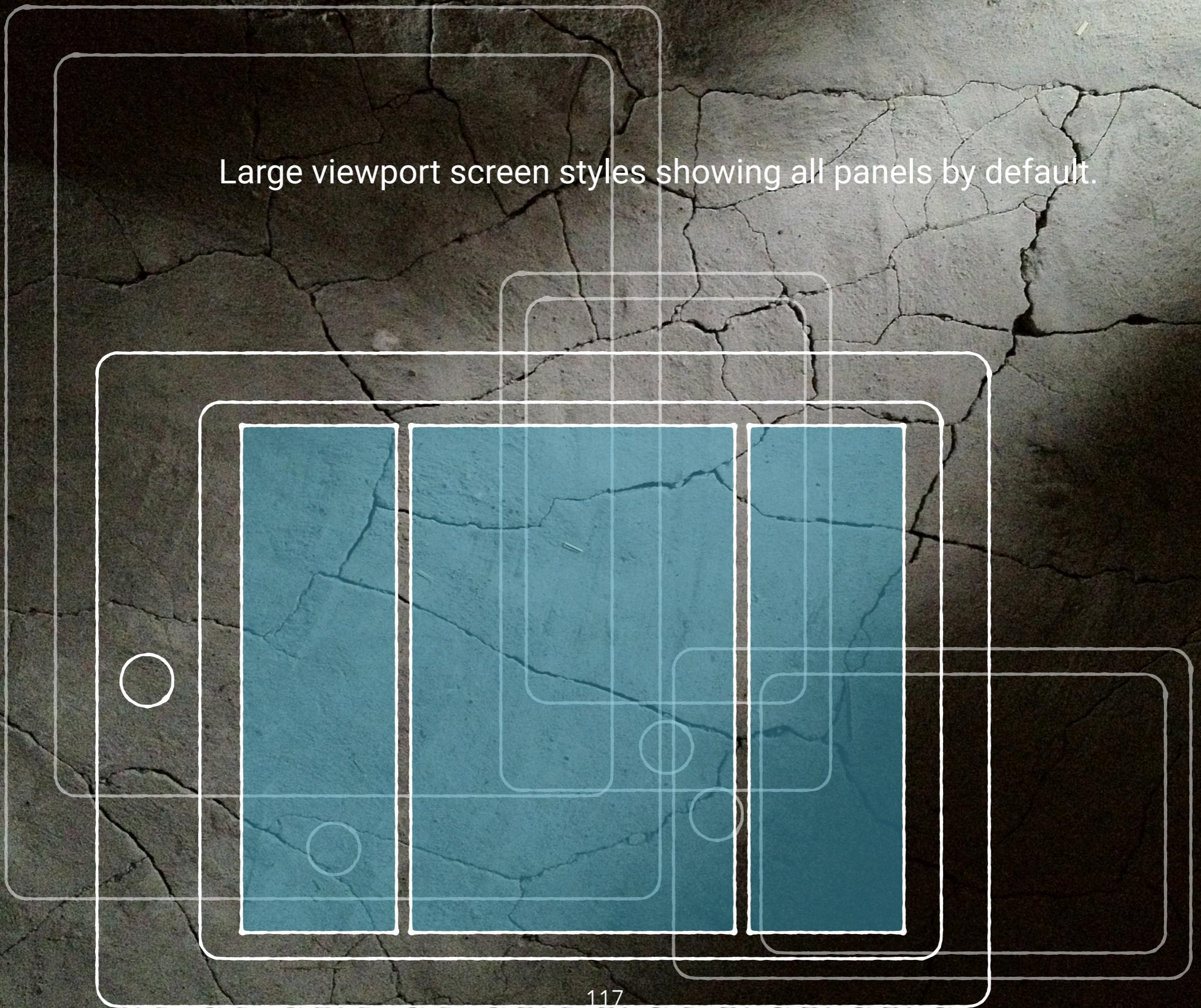
Medium viewport screen styles for access to the right.

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.

HEADER

NAVIGATION

MAIN
CONTENT

ASIDE
CONTENT

FOOTER

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 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.

#

CSS Grid Layout (level 1)  - CR

Method of using a *grid* concept to lay out content, providing a mechanism for authors to divide available space for layout into columns and rows using a set of predictable sizing behaviors. Includes support for all `grid-*` properties and the `fr` unit.

Current aligned

Usage relative

Date relative

Filtered

All



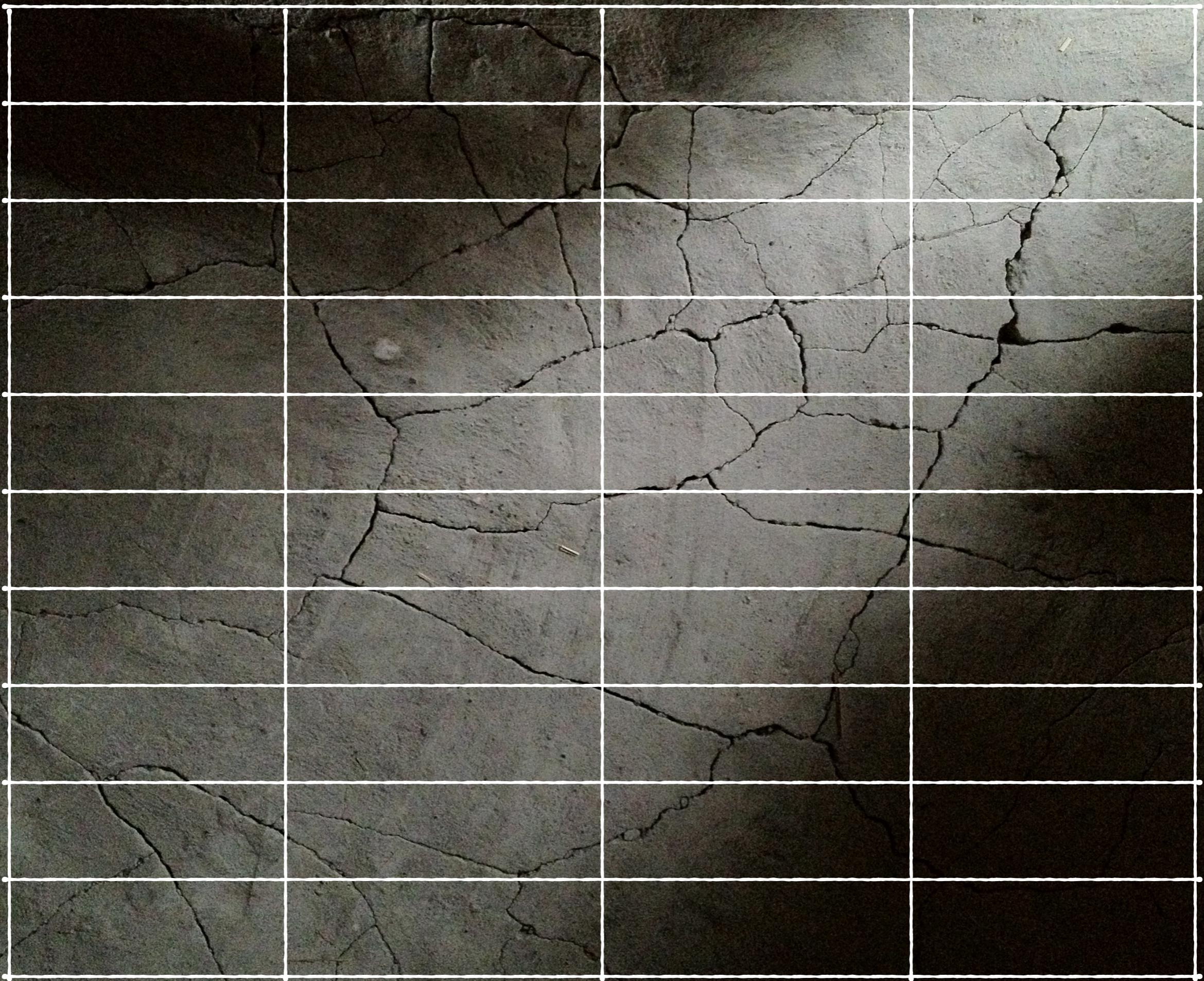
IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *
		2-39	4-28			
		3 40-51	1 29-56		10-27	
6-9	2 12-15	4 52-53	4 57	3.1-10	1 28-43	3.2-10.2
2 10	16-84	54-80	58-84	10.1-13.1	44-70	10.3-13.7
2 11	85	81	85	14	71	14
		82-83	86-88	TP		

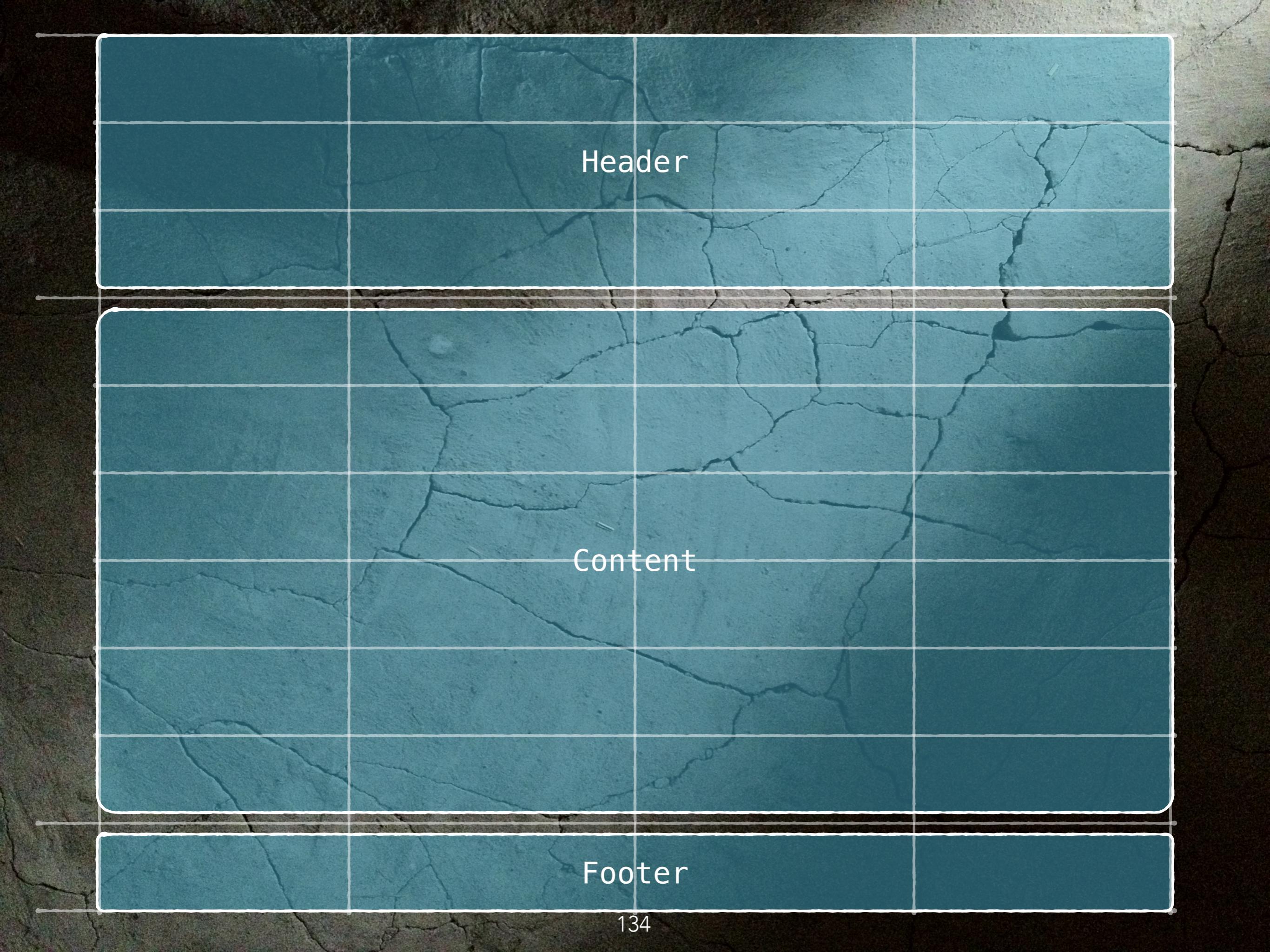
FLEX CONTAINER

```
<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

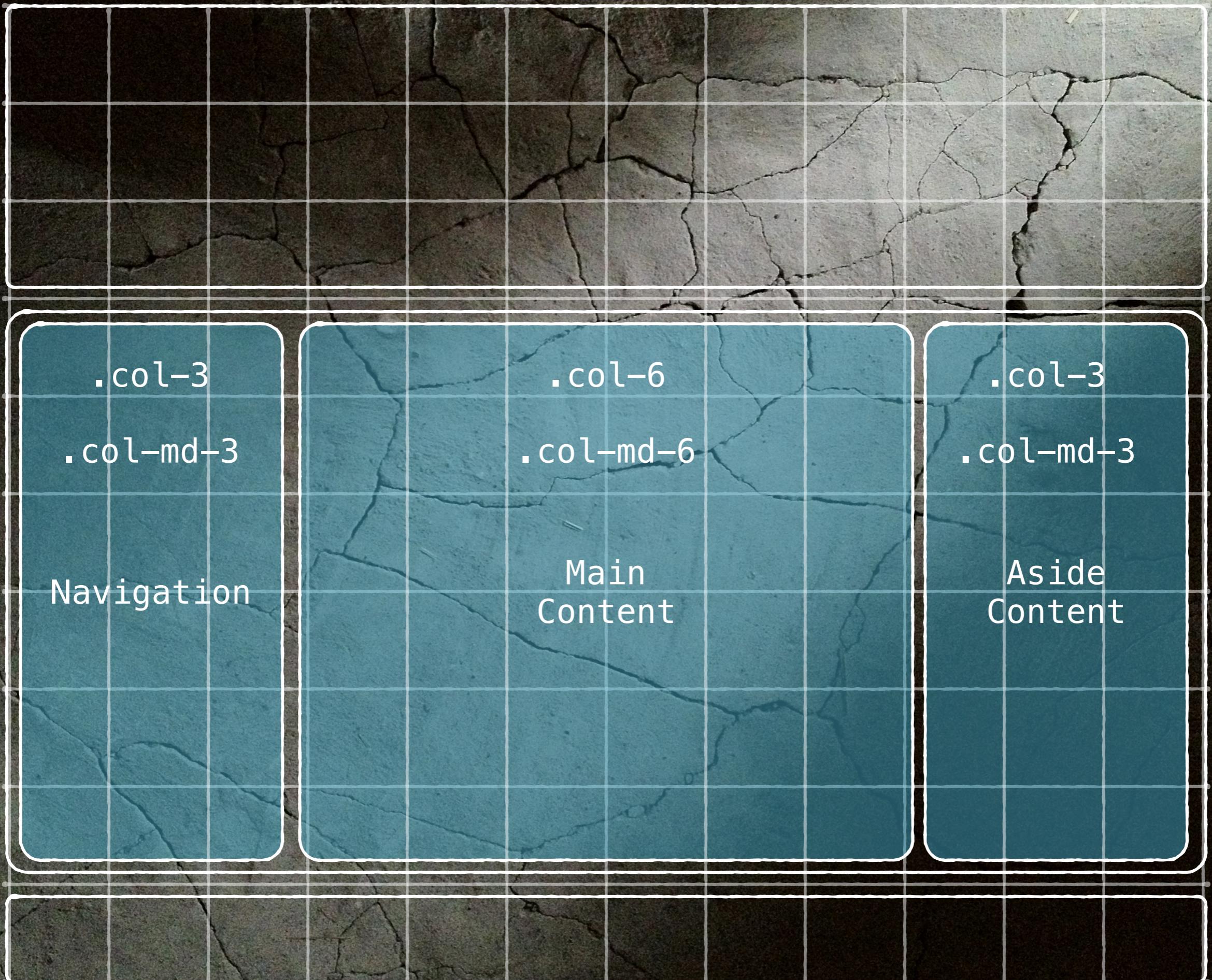
Content

Footer

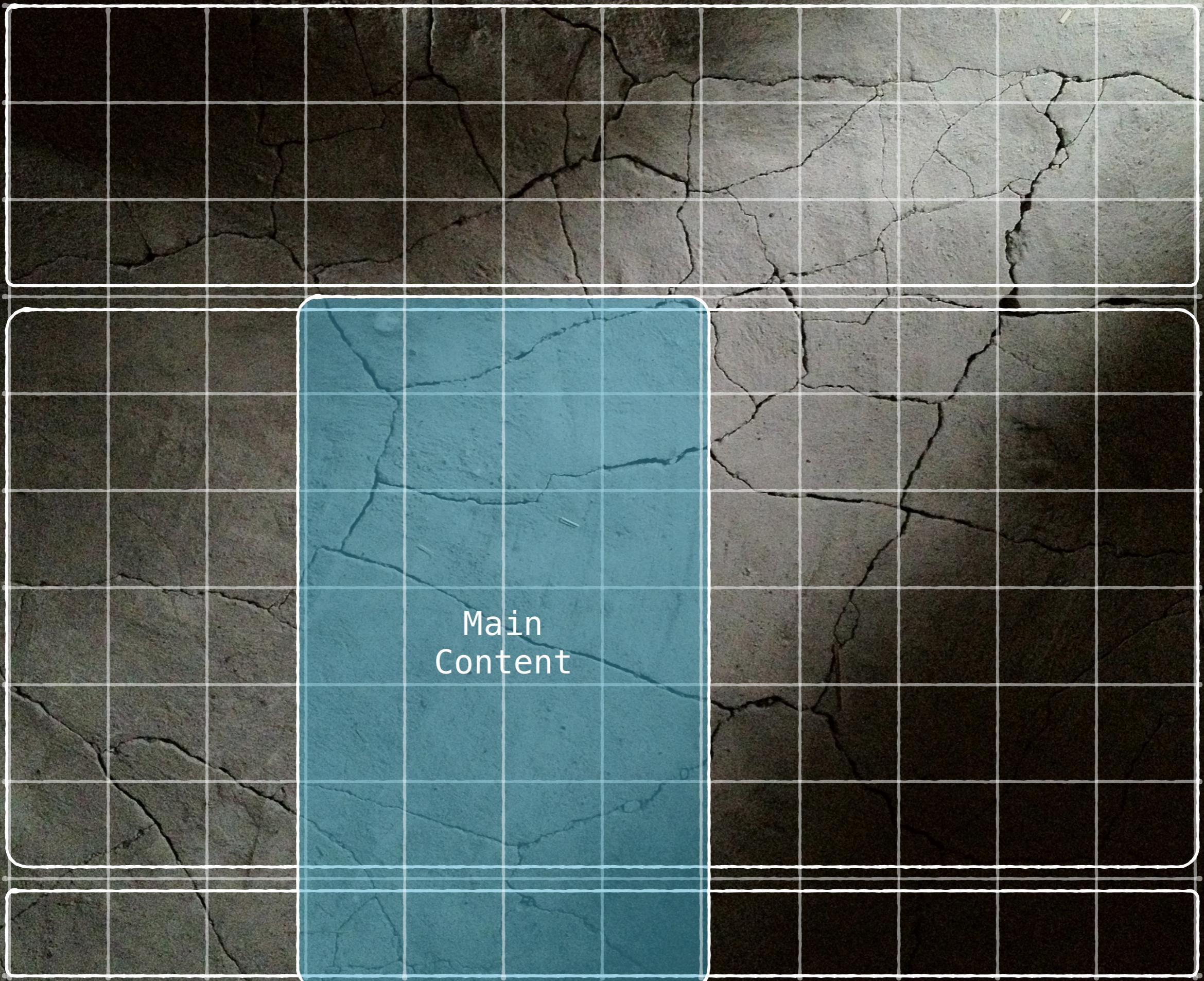
FLEX ITEMS

```
.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;  
}
```

12 columns = 960px



12 columns = 960px



12 columns = 960px

.col-3
.col-md-3

.col-6
.col-md-6

.col-3
.col-md-3
.col-sm-12

Navigation

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