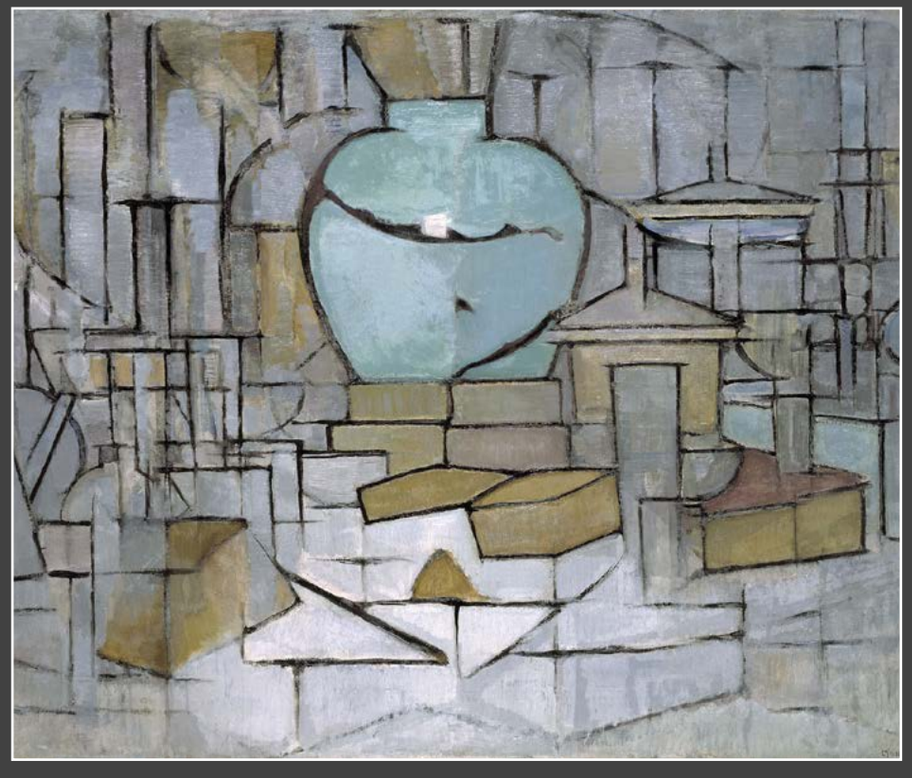
HTML5 & CSS

De Basis van Positioning

DE BASICS: NORMAL FLOW
LEARN THE RULES

so you can break them



NORMAL FLOW *(of: static)*

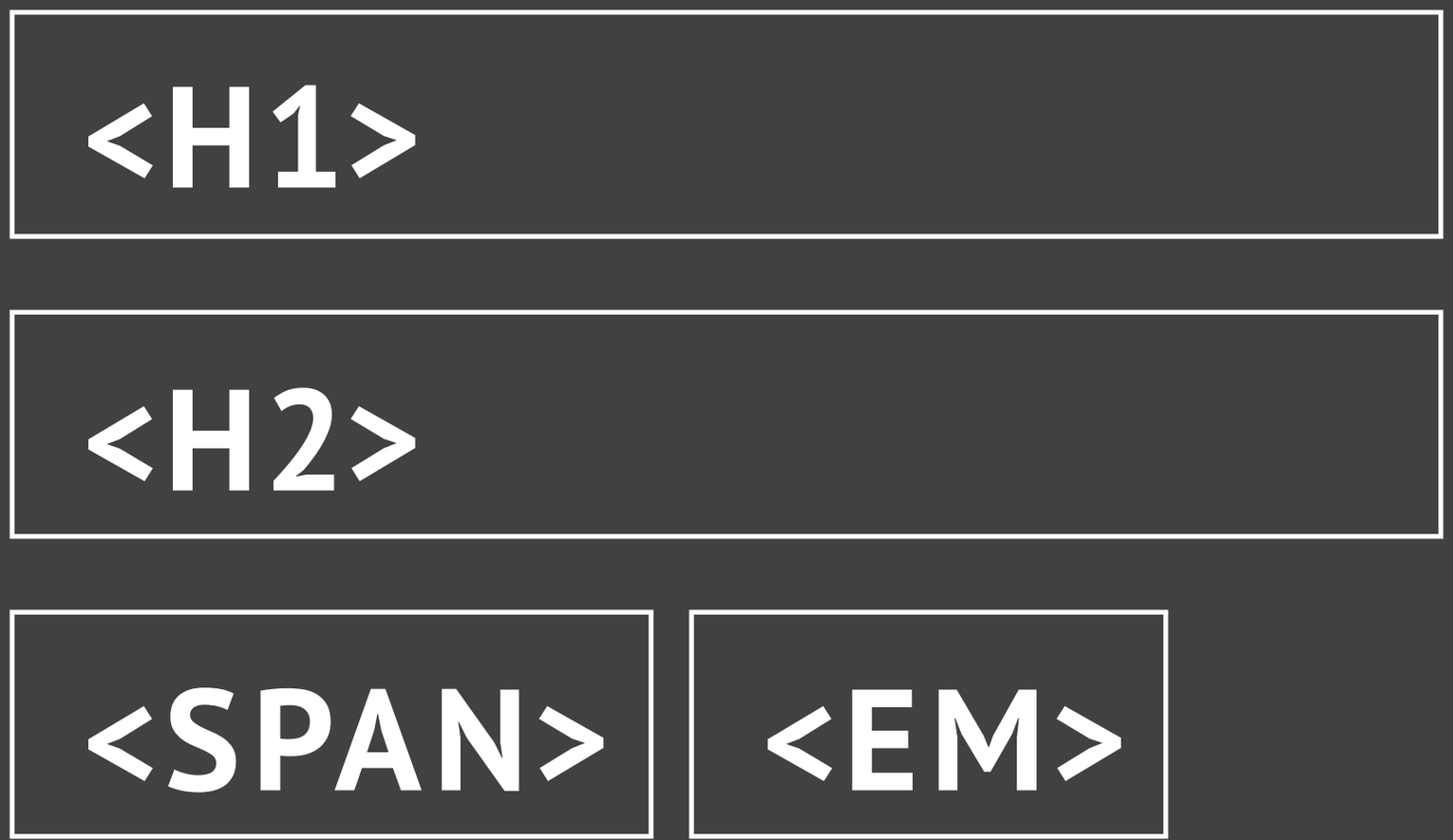
alle elementen op elkaar

- *Volgt de volgorde van schrijven in je HTML*
- *Het eerste element komt bovenaan links in de browser*
- *Het volgende element komt **ernaast** of **eronder***

https://codepen.io/petervandenheuvel/pen/xyYXww?editors=1100

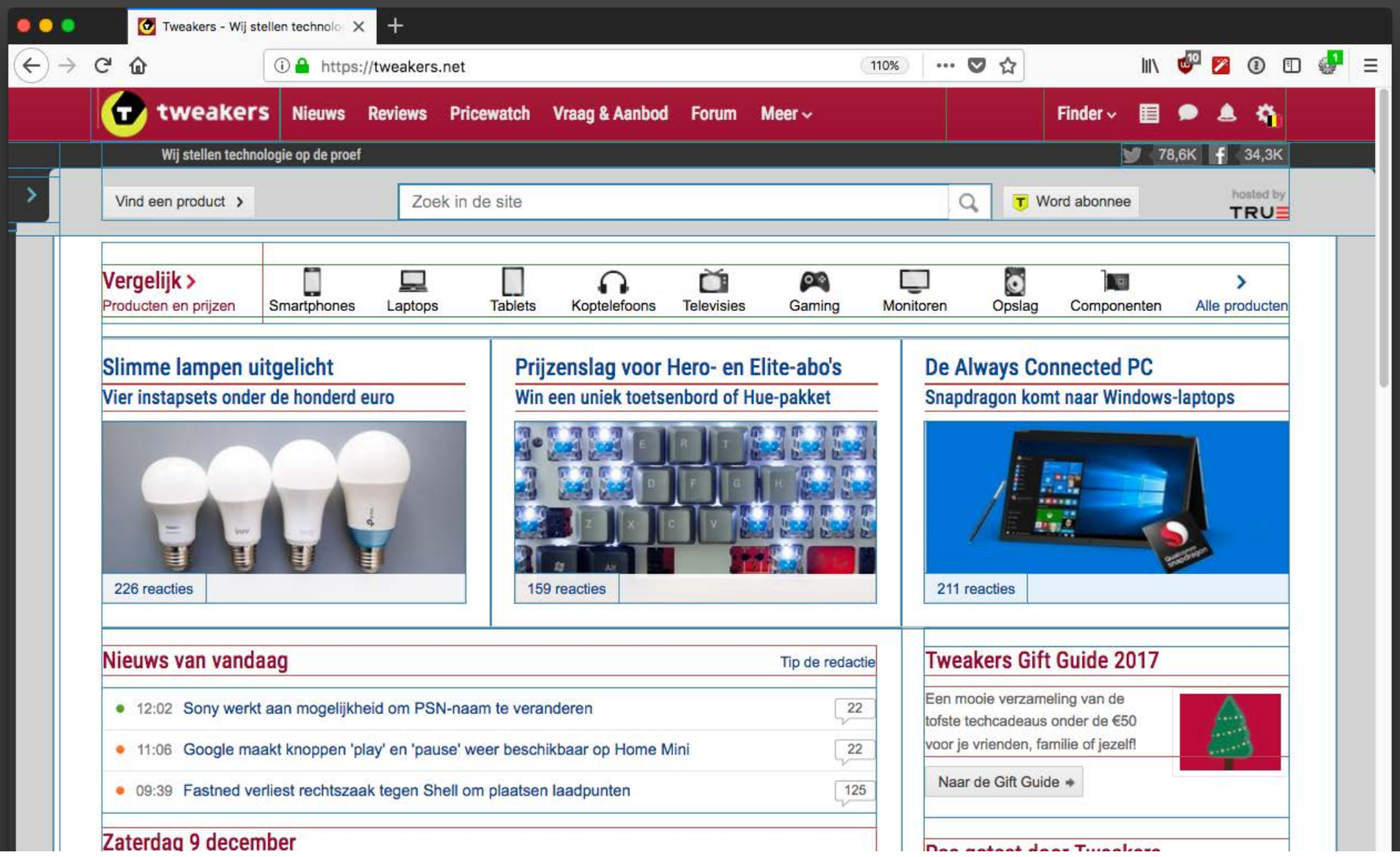
BLOCK VS INLINE

dozen, overal dozen!



MAAK DOZEN ZICHTBAAR

gebruik pesticide.io of web-dev toolbar



NORMAL FLOW

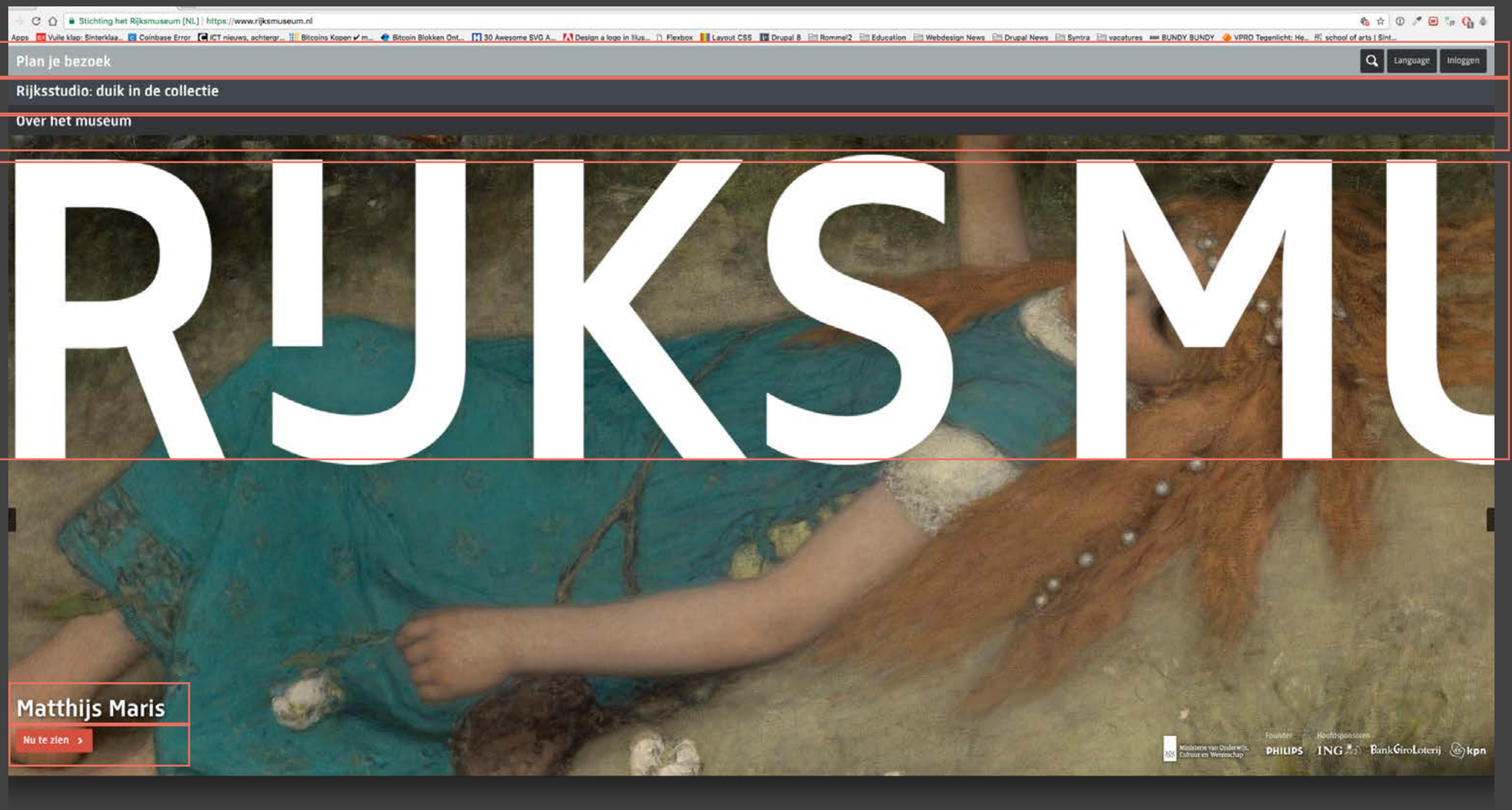
'verschuiven' binnen normal flow:

```
div.element-in-normal-flow{
    margin-left: 20px;  /* Move left outside box */
    padding-left: 40px; /* Move left inside box */

    display: block; /* Change to block-level */
    display: inline; /* Change to inline-level */
    display: inline-block; /* Change to hybrid */
}
```

Let op: *- Gebruik kleine waarden om elementen te verplaatsen:*
(een margin-top: 500px of margin-left: -250px is te vermijden)

SOMS MAG HET COMPLEXER

Alle complexe layouts zijn ook 'dozen op elkaar'



DE BASICS:

Hoe verschuif je de elementen uit hun natuurlijke flow?

1. **RELATIEF POSITIONEREN**
2. **ABSOLUUT POSITIONEREN**
3. **FIXED POSITIONEREN**
4. **FLOATEN**

GEEN POSITIONING

normal flow of position: static

static.html — UNREGISTERED

static.html

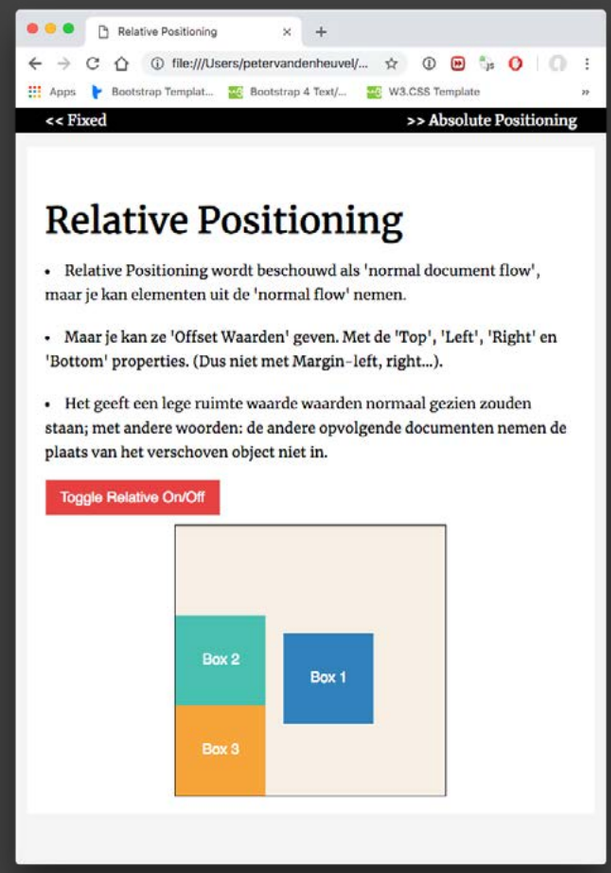
```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Static</title>
</head>
<body>
    <h1>Header 1</h1>
    <style>
    h1{
        position: static;
    }
    </style>
</body>
</html>
```

In css-advanced-2018 on No, File is untracked, Line 1, Column 3 — Spaces: 4 — HTML

1) RELATIVE POSITIONING

h1{ position: relative;}

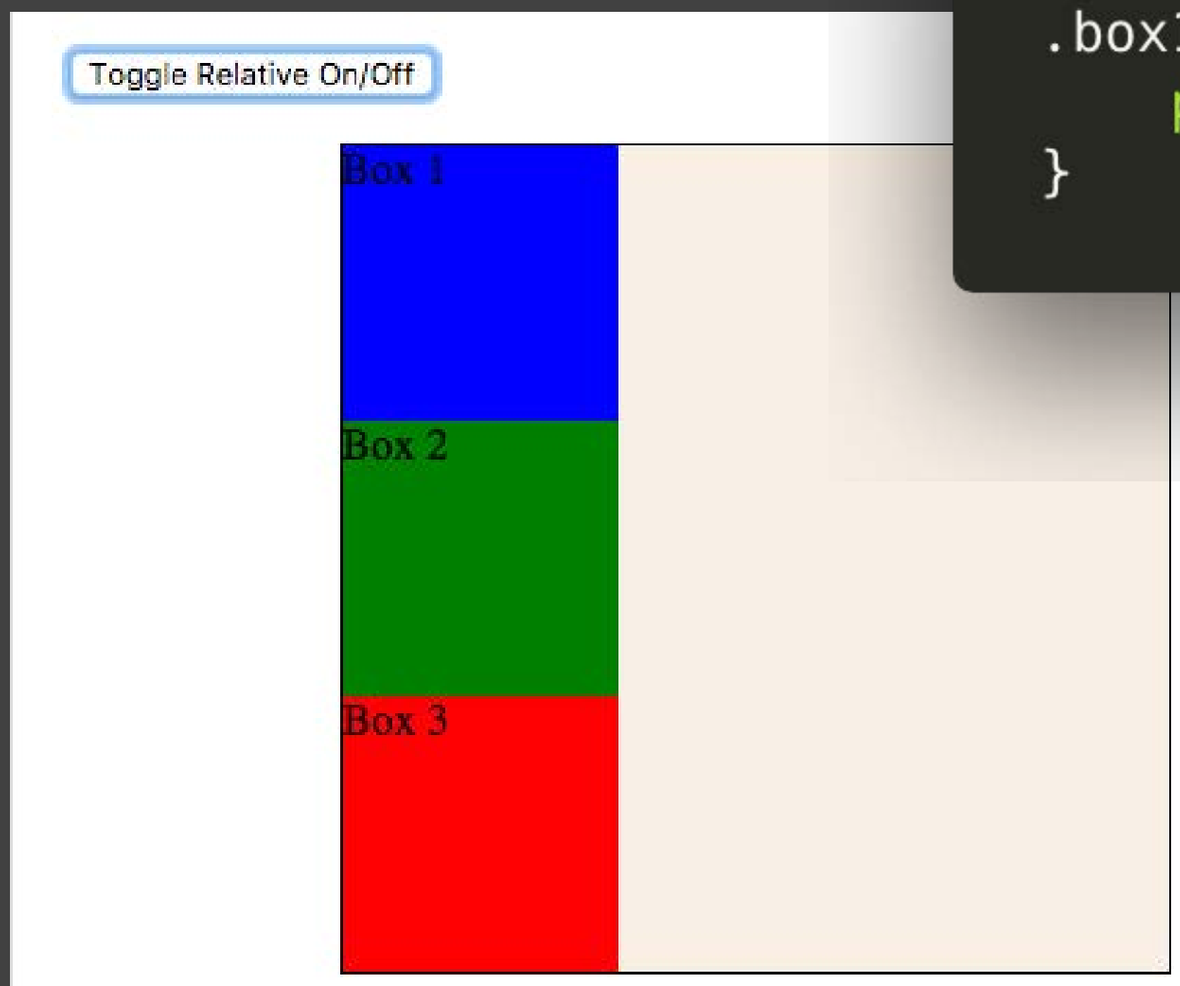
1) Het geselecteerde element gedraagt zich als een static (niet-positioneerd) element.
2) Totdat je het een positie geeft. (Offset waarden)
3) Ofwel: het element blijft in de normal flow.



VOORBEELD 1

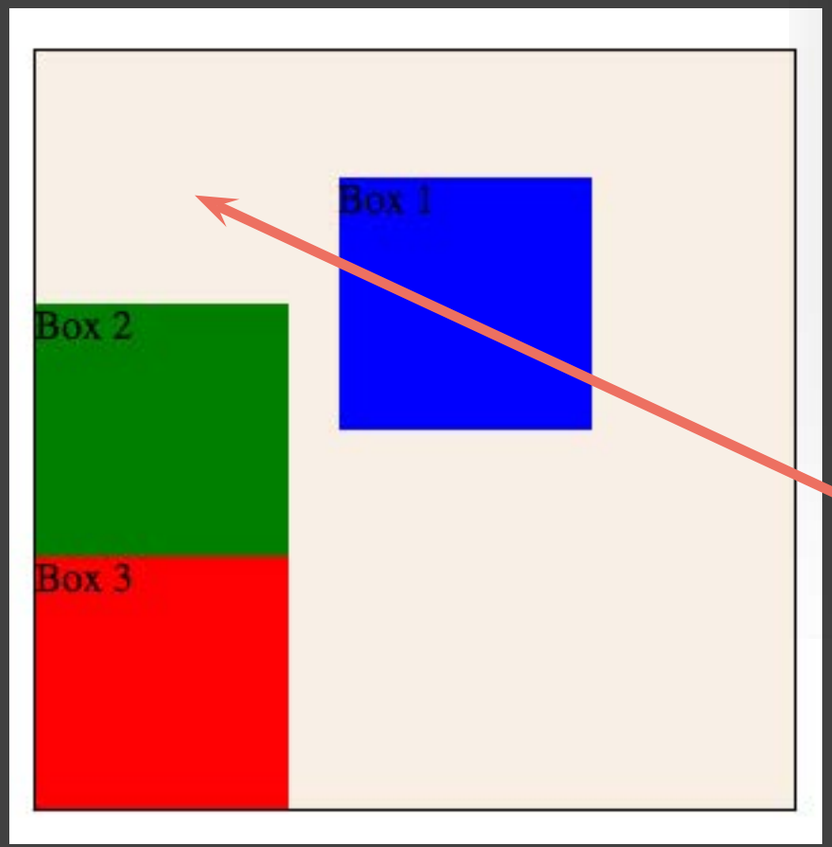
er lijkt niets veranderd:
maar doos 1 heeft nu 'position:relative'.

```
.box1 {
    position: relative;
}
```



VOORBEELD 2

totdat je offset waarden geeft:




```
.box1 {
    position: relative;
    left: 10px;
    top: 10px;
}
```

De ‘geest’ van .box1 blijft staan op de normale plek.

https://codepen.io/petervandenheuvel/pen/gBveyb

1) RELATIVE POSITIONING

Het volgende element wordt gepositioneerd op de plaats waar het normaal gezien zou voorkomen in normal flow.

De normal flow is exact hetzelfde voor de verplaatsing,
ookal is het element visueel verplaatst.

Gevolg: JE MOET WETEN WAAR HET ELEMENT IN NORMAL FLOW ZOU STAAN!
Extra MARGIN toevoegen maakt dit nog complexer, pas daar mee op.

OPDRACHT

maak het voorbeeld na en test de gevolgen van:
'position: relative'.

1) Maak het kader: een div.container: width & margin: 0 auto;

2) Maak de drie kubussen (.box) daarin (block level-elementen..)

3) Gebruik een klasse .box om hem te verkleinen: 100px op 100px.

4) Test het toepassen van pos: relative op de eerste box:
   - Wat gebeurt er met .box1 ?
   - Wat gebeurt er als je top en left-values toevoegd?
   - Hoe reageren de andere boxes?
   - Klopt dit met wat je verwacht had?

```css
.container{
  background: lightgrey;
  width: 50%;
  margin: 0 auto;
}

.box{
  width: 100px;
  height: 100px;
}

.one{background:blue;}
.two{background:green;}
.three{background:red;}

.one{
  position: relative;
  left: 125px;
  top: 50px;
}
```

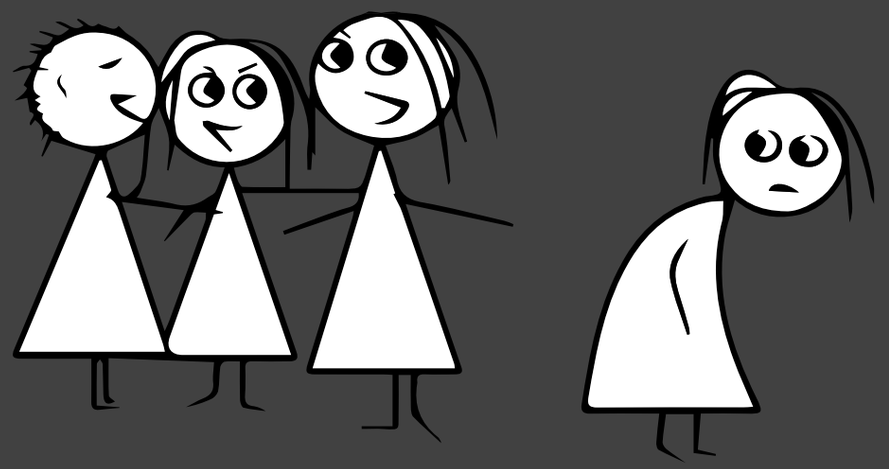
❷ ABSOLUUT POSITIONEREN

ABSOLUTE POSITIONING

```
.box1{
    position: absolute;
}
```

- *Het 'x' element wordt **UIT** de '**natuurlijke flow**' van de pagina gehaald.*
- *Het volgende element in de normal flow doet dus net alsof x niet bestaat.*

Als je iets position: absolute geeft, doen alle andere elementen alsof dit element niet meer bestaat op de pagina: 'uit de flow'-nemen.



ABSOLUTE POSITIONING

Opdracht: pas je (vorige) voorbeeld aan:

HTTPS://CODEPEN.IO/PETERVANDENHEUVEL/PEN/ZMRJOM

- *Het verplaatst zich ten opzichte van het eerste (parent) element dat postioning heeft, als dat er niet is neemt hij de <body>-tag.*

(Dit werkt alleen als je element 'left, right, top, bottom'-waarden heeft...)

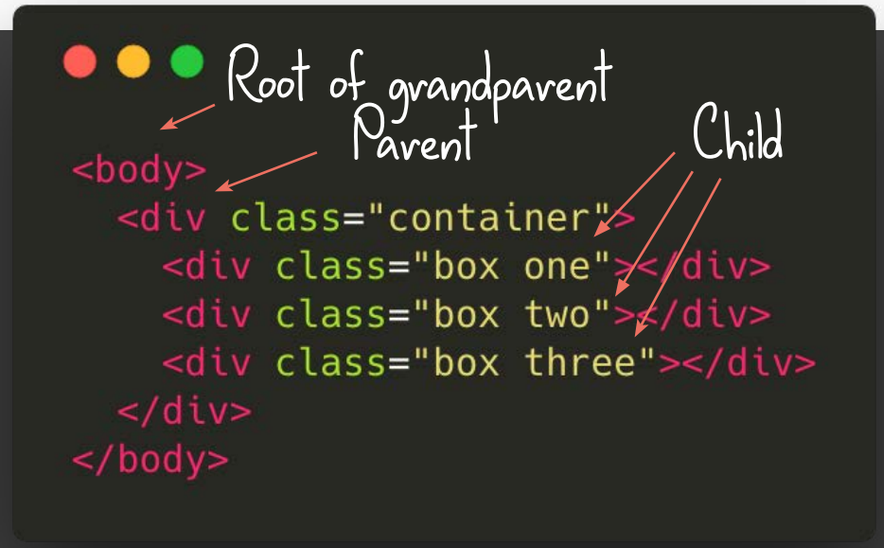
Het lijkt dan alsof het element position: fixed heeft...

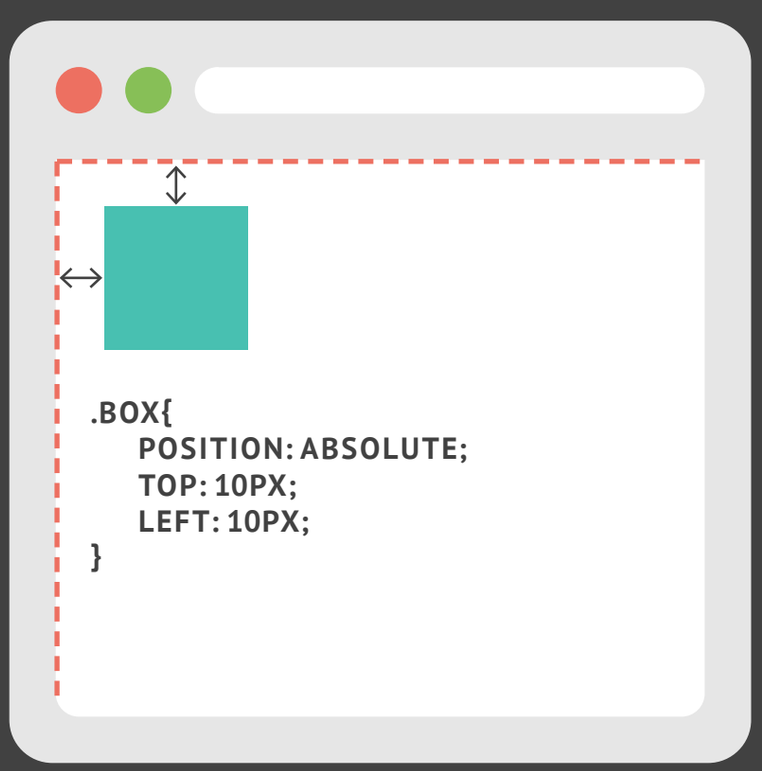
```css
.container{
  background: lightgrey;
  width: 50%;
  margin: 0 auto;
}

.box{
  width: 100px;
  height: 100px;
}

.one{background:blue;}
.two{background:green;}
.three{background:red;}

.one{
  position: absolute;
  left: 125px;
  top: 50px;
}
```

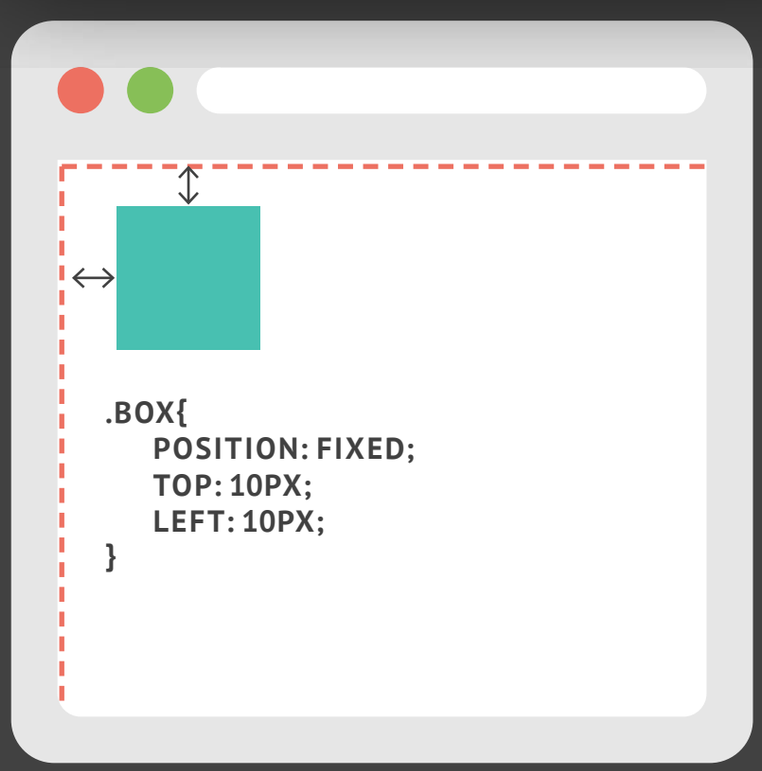
ABSOLUTE POSITIONING

- *Het verplaatst zich ten opzichte van het eerste (parent) element dat postioning heeft, als dat er niet is neemt hij de <body>-tag.*

  Het lijkt dan alsof het element position: fixed heeft...



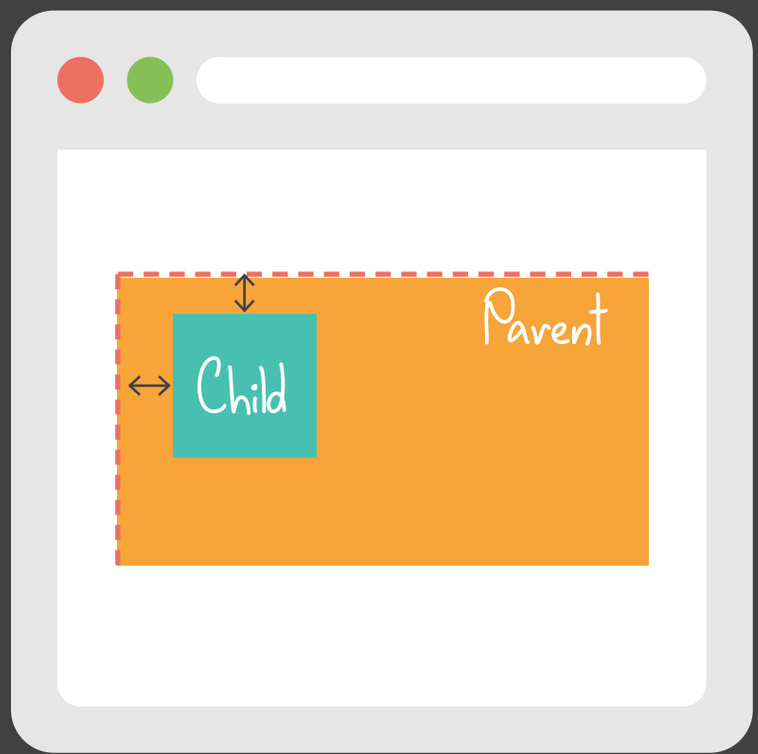




ABSOLUTE POSITIONING

- *Als er wel een parent is met positioning (bijv. 'relative');*
  dan plaatst het absoluut gepositioneerde kind zich t.o.v. de parent.

```
Parent
.orange {
    position: relative;
}
    Child
.green {
    position: absolute;
    top: 10px;
    left: 10px;
}
```



ABSOLUTE POSITIONING

De breedte van het element

Vanaf het moment dat je een element 'Position: absolute' geeft,
gaat deze naar de minimale breedte van de zijn inhoud.
Is er geen inhoud, dan wordt deze '0'.

Je kan dat eens testen in het vorige voorbeeld door de 'width' weg te halen,
en de inhoud (text) te verwijderen.

PRAKTISCH VOORBEELD

absolute positioning


```
.BLAUW-BUTTON {
  POSITION: ABSOLUTE;
  BOTTOM: 10PX;
  RIGHT: 10PX;
}
```

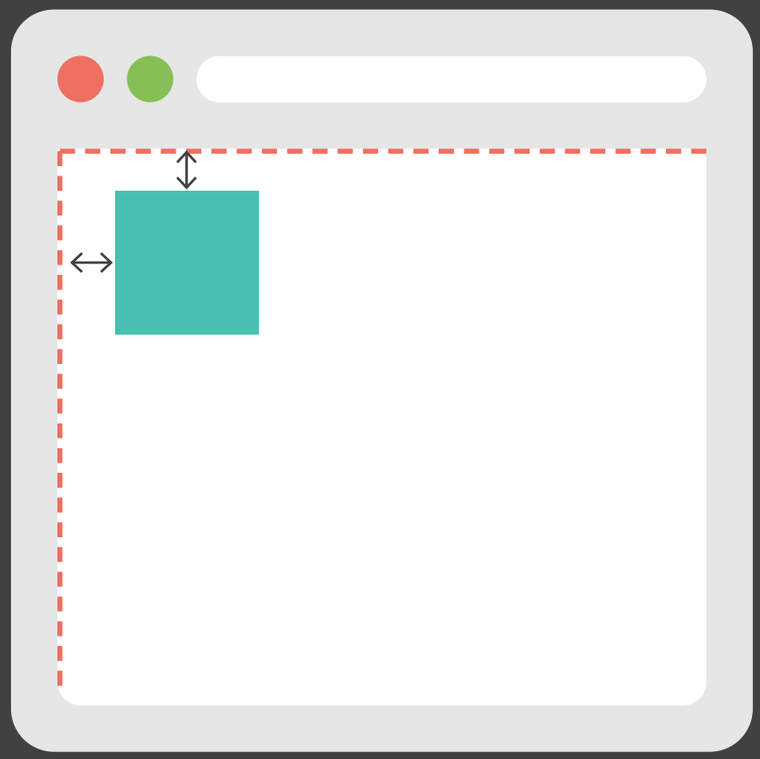
```
.KADER {
  POSITION: RELATIVE;
}
```

3 FIXED POSITIONEREN

FIXED POSITIONING

het element wordt geplaatst t.o.v. de <body>

```
.GREEN {
  POSITION: FIXED;
  TOP: 10PX;
  LEFT: 10PX;
}
```



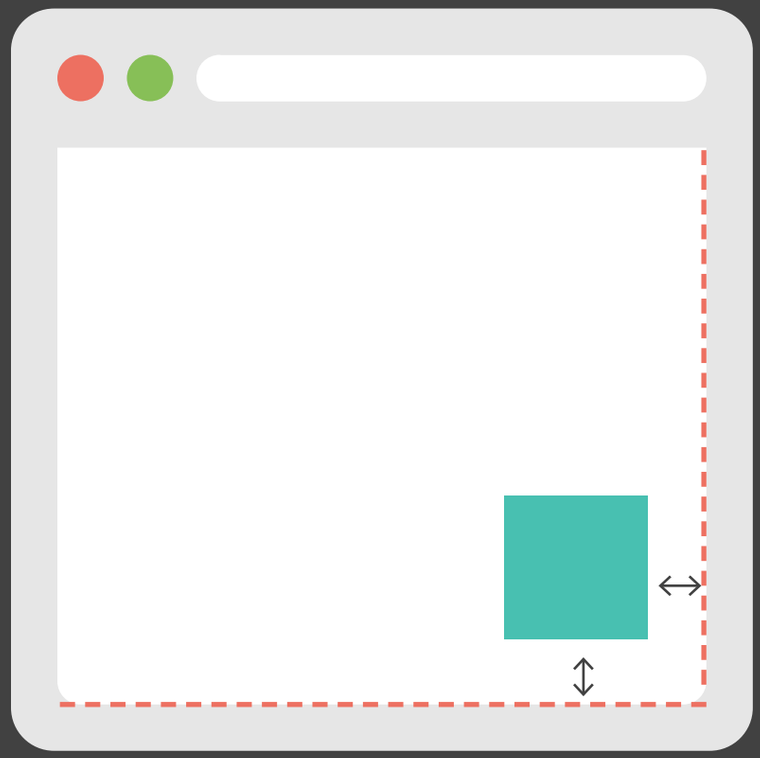
FIXED POSITIONING

het element wordt geplaatst t.o.v. de <body>

```
.GREEN {
  POSITION: FIXED;
  BOTTOM: 10PX;
  RIGHT: 10PX;
}
```

HET ELEMENT BLIJFT ALTIJD ZICHTBAAR IN HET SCHERM, OOK ALS JE SCROLLED.

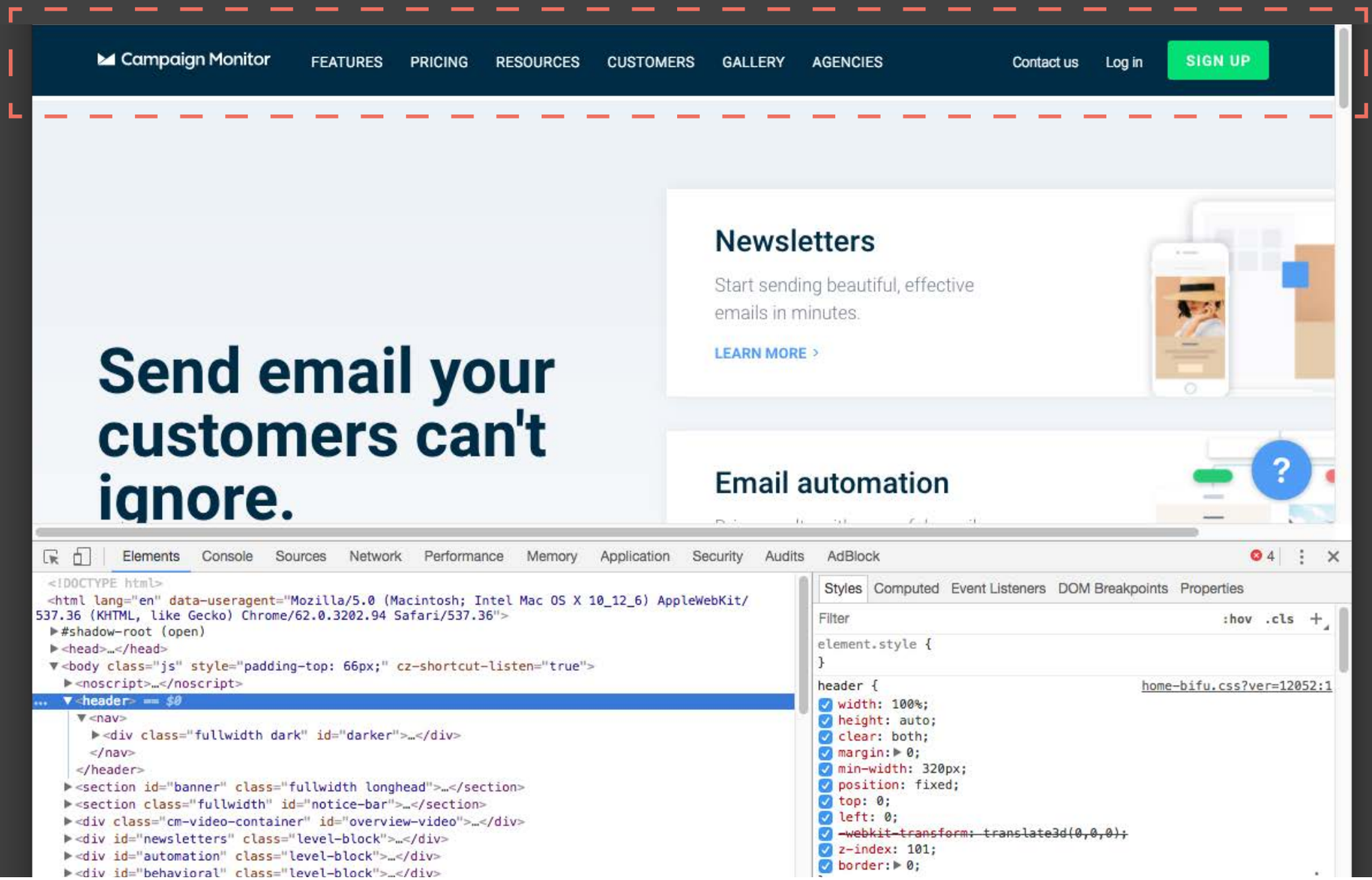
IDEAAL VOOR IETS DAT ALTIJD ZICHTBAAR MOET ZIJN: HEADER.



POSITION: FIXED

Praktisch voorbeeld

Voorbeeld van een website: een fixed header die plakt



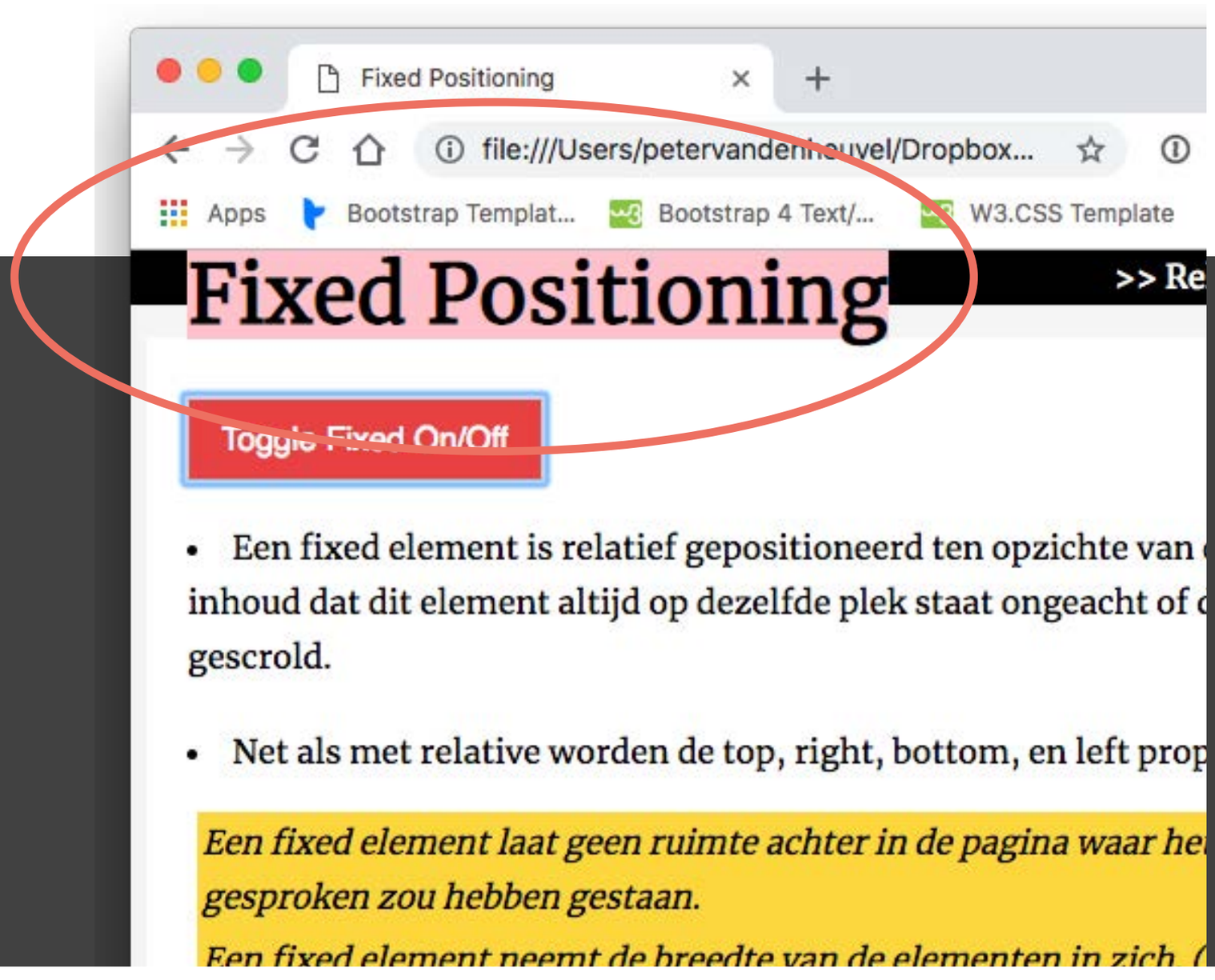
FIXED POSITIONING

*het element wordt **uit** de flow gehaald.*

Het volgende element gedraagt zich alsof het niet bestaat.

(Net als bij absolute positioning)

```
.HEADER {
  POSITION: FIXED;
  TOP: 0PX;
  LEFT: 0PX;
}
```



FIXED POSITIONING

De breedte van het fixed-element wordt net zo breed als de inhoud. (Exact zoals bij absolute positioning)

Bij het voorbeeld van de header, zal je dat zo moeten oplossen, anders is de header niet even breed als het venster:

```
.HEADER {
  POSITION: FIXED;
  TOP: 0PX;
  LEFT: 0PX;
  WIDTH: 100VW OF 100%;
}
```

OVERLAPPING

beinvloeden

```
.TOP {
  POSITION: RELATIVE;
  Z-INDEX: 1;
}
```

```
.BOTTOM {
  POSITION: RELATIVE;
  Z-INDEX: 0;
}
```

Z-INDEX

Gebruik z-index om de stapelvolgorde van gepositioneerde elementen te beinvloeden.
Let op: 'gepositioneerde elementen', anders werkt het niet.
Dat wil zeggen; alles behalve 'static' dus.

Hoe hoger de waarde; hoe dichterbij de kijker.
999 is bovenop de stapel, 0 is onderop. Of -2, nog dieper.

```
.GREEN {
  POSITION: RELATIVE;
  Z-INDEX: 1;
}
```

```
.RED {
  POSITION: RELATIVE;
  Z-INDEX: 2;
}
```

Z-INDEX

alle elementen op elkaar

- *Niet-gepositioneerde elementen (position:static) hebben **GEEN** z-index, ofwel een z-index van 0.*
- *Bij een **gelijke** z-index komt het **laatste** element in de broncode (HTML) hoger te liggen in de stapelvolgorde.*

4 FLOATS

```
.YOU-WILL-FLOAT-TOO {
  FLOAT: LEFT;
}
```

```
.BILL {
  CLEAR: BOTH;
}
```

FLOATS

alle elementen op elkaar

- *Floats worden nog het meest gebruikt om kolommen-layouts te maken.*
- *Dat komt vanwege de historisch slechte ondersteuning van inline-block*
- *Sowieso: inline-block voor layout; world of pain: https://iamsteve.me/blog/entry/inline_block, don't go there…*

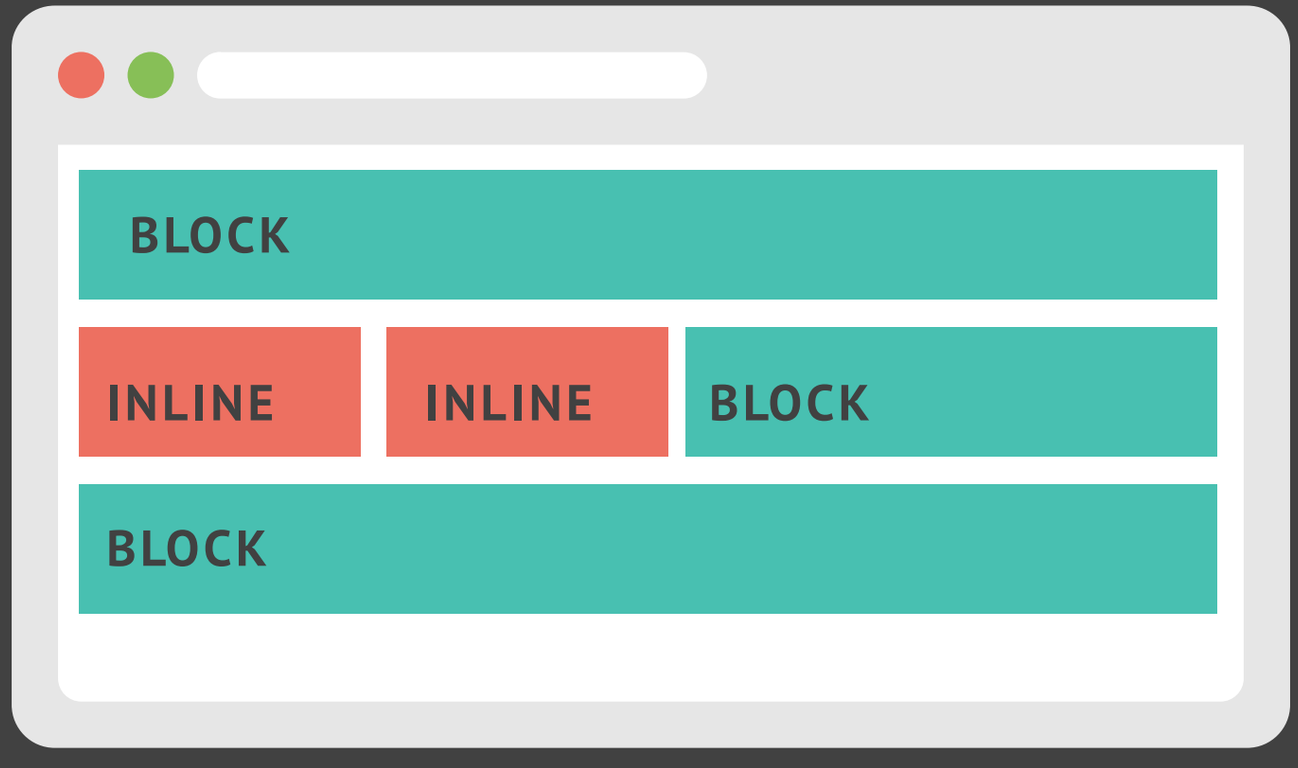
Maar eerst terug naar de basics:

BLOCK VS INLINE

terug naar de basics:

```
<div class="parent">
    <p>Block element 1.</p>
    <p>Block element 2. Inline element <span>here</span>.</p>
    <p>Block element 3.</p>
</div><!-- /.parent -->
```

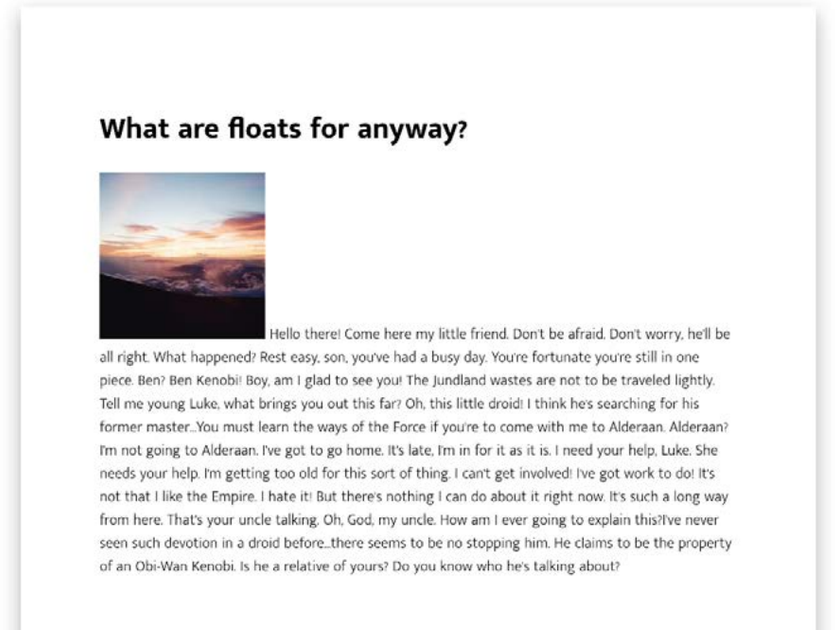
- *Standaard stapelen block-levels op elkaar, en zijn ze 100% breed. In volgorde van schrijven in de HTML.*
- *Inline elementen blijven in hun context, nemen de breedte die ze nodig hebben.*
- *Floats halen ze niet uit de natural flow, maar drijven naar de top-left of top-right van hun container.*
  (zie voorbeeld met image-float)

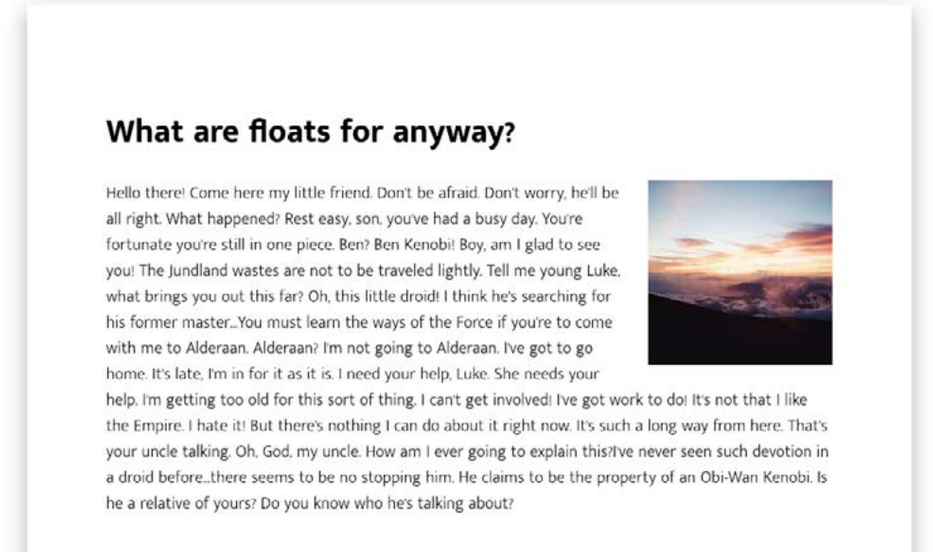


KLASSIEKE FLOAT:

een afbeelding in een stuk tekst

```
img {
  float: right;
}
```





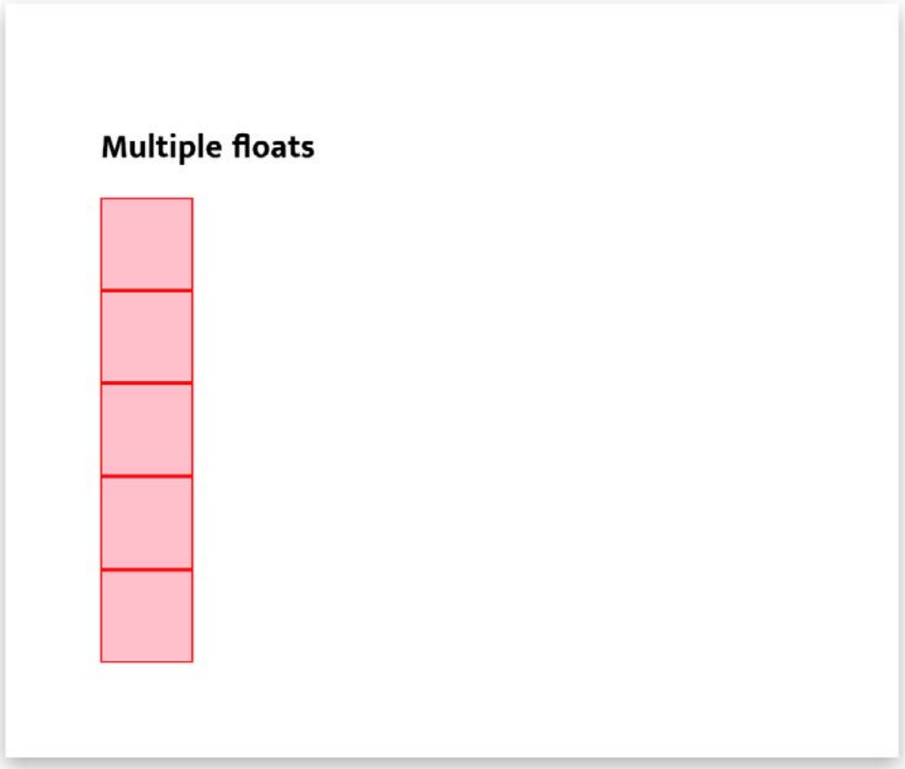
De afbeelding staat in de tekst (inline)

De tekst vloeit langs de afbeelding.

https://codepen.io/petervandenheuvel/pen/gqxoRZ?editors=1100

MULTIPLE FLOATS

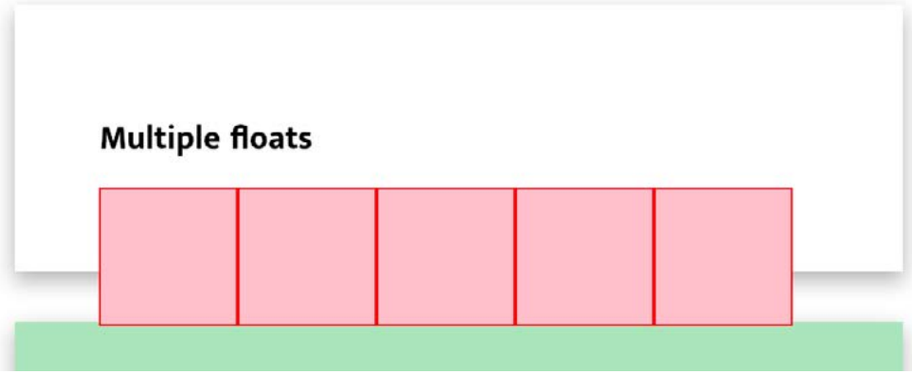
waarom mensen zoveel 'height' gebruiken:



-Een witte met een slagschaduw

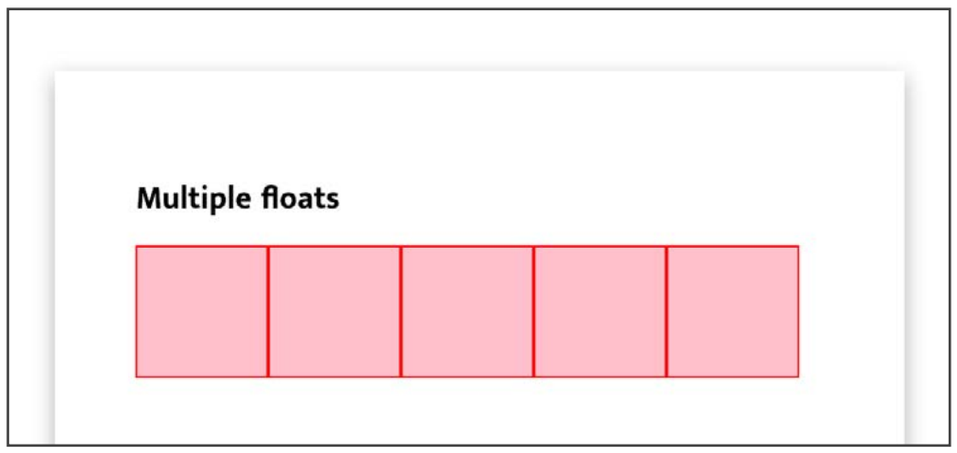
- rode blokjes zijn .

- Allemaal 'float: left' geven...



De parent (witte div) respecteert de hoogte van de rode blockjes niet.

Kan je oplossen met deze een min-height te geven.



LAYOUT MET KOLOMMEN

Nu wil je drie kolommen: 30%

We hebben een parent: *section.float-layout-one*

3 child 's met een klasse .col

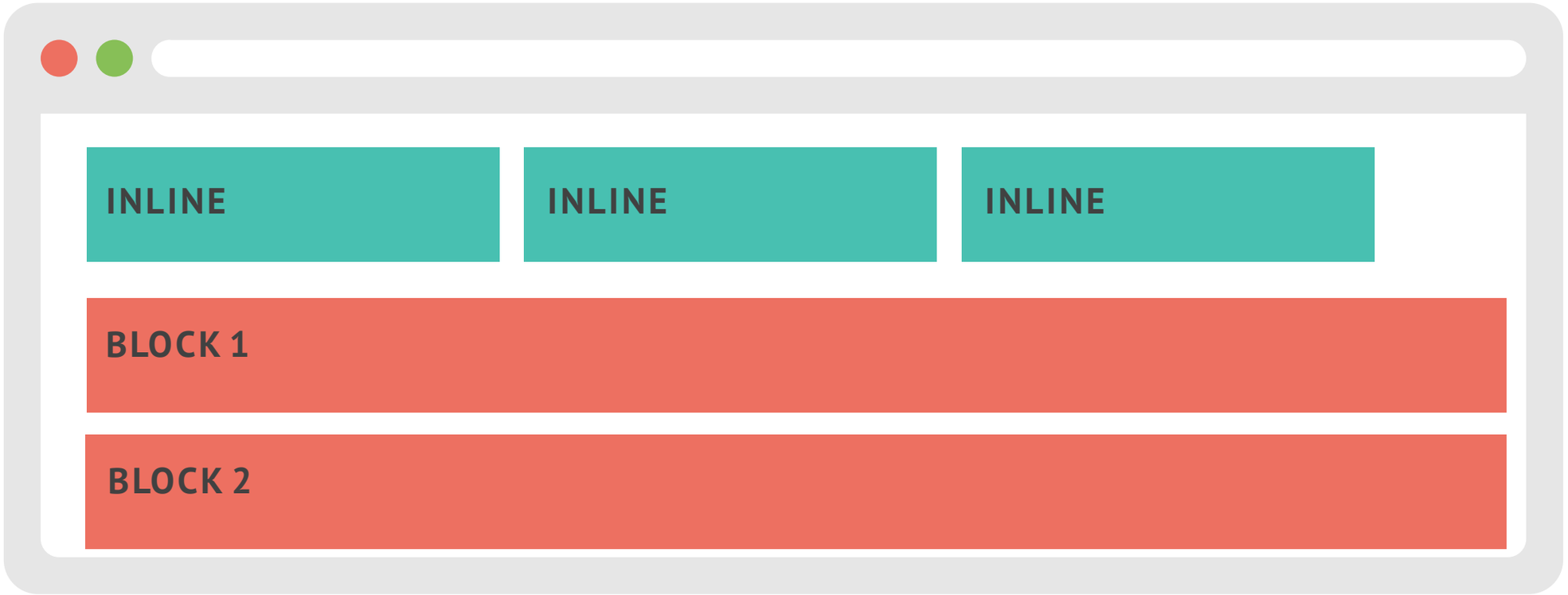
Het meest logische is nu simpel: float: left.

Dan krijg je dit:


```
<section class="float-layout-one clearfix">
  <h1>Floats for layouts</h1>
  <div class="col">
    <p>Hello there! Come here my little friend. Don
happened? Rest easy, son, you've had a busy day. You
Boy, am I glad to see you! </p>
  </div>
    <div class="col">
    <p>Hello there! Come here my little friend. Don
happened? Rest easy, son, you've had a busy day. You
Boy, am I glad to see you! </p>
  </div>
    <div class="col">
    <p>Hello there! Come here my little friend. Don
happened? Rest easy, son, you've had a busy day. You
Boy, am I glad to see you! </p>
  </div>
</section>

<style>
.float-layout-one {
  background: #a9e5bb;
}
.float-layout-one .col {
  width: 30%;
  margin-right: 3%;
  float: left;
}
</style>
```

FLOAT: CLEAR

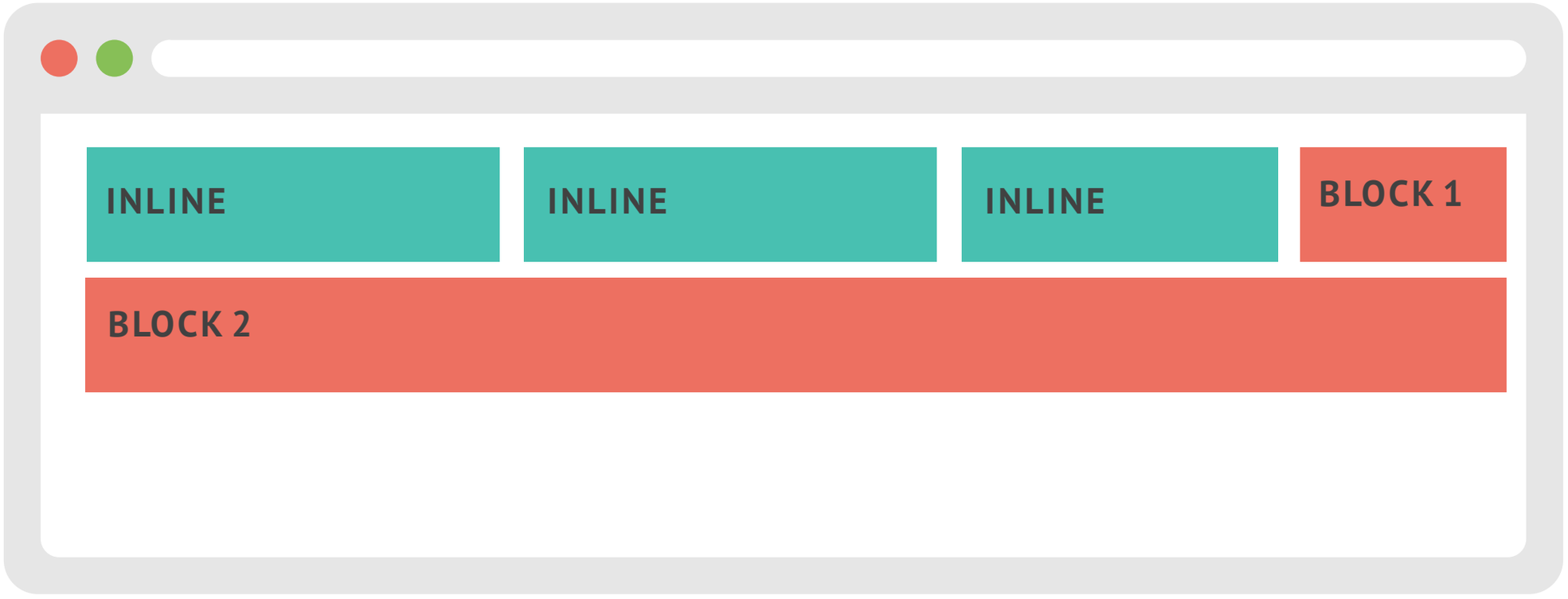
Vraag: wat gebeurt er als je 'Block 1' float?



HTTPS://CODEPEN.IO/PETERVANDENHEUVEL/PEN/JXOPZW

FLOAT: CLEAR

Vraag: wat gebeurt er als je 'Block 1' float?



CLEAR THE FLOAT

waarom clearfix handig is

*De eerste oplossing wat **na de floated elements** een <br style=”clear: both”> toevoegen.*

*Dit gaf enkel een nieuwe lijn, maar je moest wel iedere keer een
 toevoegen. Lastig.*

Rond 2008 komt iemand op het idee om daar het pseudo-element ‘:after’ voor te gebruiken in combinatie met de ‘content’ -property.

Die ‘hack’ zie je hier naast en was meer als 10 jaar ‘standaard’.

Oa. Twitter Bootstrap 2 & 3 gebruiken dit.

```
<section class="float-layout-one clearfix">
  <h1>Floats for layouts</h1>
  <div class="col">
    <p>Hello there! Come here my little friend. Don
happened? Rest easy, son, you've had a busy day. Yo
Boy, am I glad to see you!
  </div>
    <div class="col">
    <p>Hello there! Come h
happened? Rest easy, son,
Boy, am I glad to see you!
  </div>
    <div class="col">
    <p>Hello there! Come h
happened? Rest easy, son,
Boy, am I glad to see you!
  </div>
</section>

<style>
.float-layout-one {
  background: #a9e5bb;
}
.float-layout-one .col {
  width: 30%;
  margin-right: 3%;
  float: left;
}
</style>
```

```
<div class="parent clearfix">
  <div>float</div>
  <div>float</div>
</div>

<style>
.clearfix::after {
  display: table;
  content: '';
  clear: both;
}
</style>
```

FLOATS:

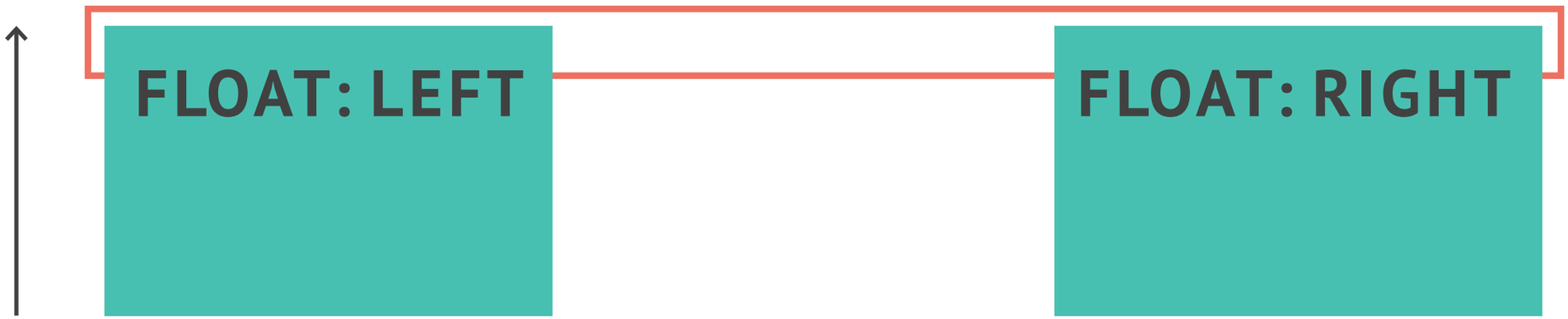
Parent elements herkennen de hoogte van floated elementen niet, en respecteren enkel de hoogte van non-floated child elements

FLOAT: LEFT

NOT FLOATED TEXT, LOREM IPSUM LOREM IPSUM LOREM IPSUM LOREM

FLOATS:

Parent elements herkennen de hoogte van floated elementen niet, en respecteren enkel de hoogte van non-floated child elements: Probleem als er verder geen inhoud is: bijv. twee kolommen layout.



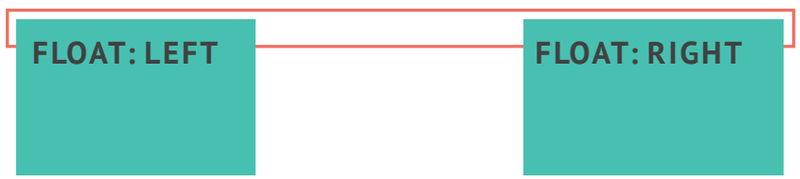
DE HOOGTE VAN DE RODE CONTAINER ZAKT NU IN TOT 0PX HEIGHT,
OF: DE PARENT COLLAPSED.

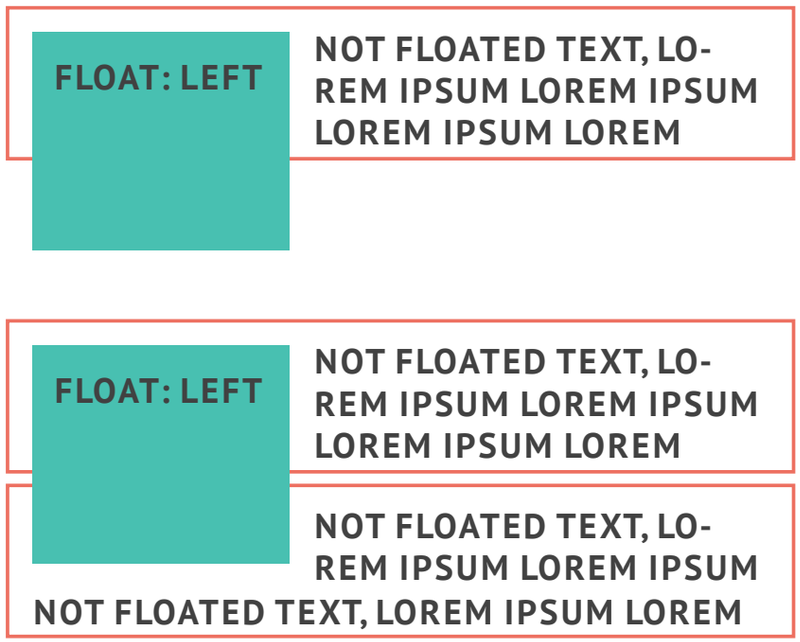
Je moet floats altijd een 'width' geven, behalve <img>, die hebben er deze van nature al.

Als een floated element geen plaats meer heeft, schuift hij op naar de volgende regel waar wel voldoen de plaats is.

CLEARING FLOATS:

Als er geen elementen zijn die onze float kunnen clearen, zoals bij het voorbeeld met de <h1>, zullen we ervoor moeten zorgen dat het vanzelf gebeurt.





When there are no elements to apply a clear to, the parent element will need to self-clear.

You'll have to force the parent to contain it's floated children...

CLEARING FLOATS:

Gelukkig zijn er 2 oplossingen:
1) Gebruik overflow met de waarden 'auto' of 'hidden'
2) Gebruik één van de clearfix-hacks.


```
.PARENT {
  OVERFLOW: AUTO;
  BORDER: 1PX SOLID RED;
}
```

```
<div class="parent clearfix">
    <p>floated element</p>
    <p>floated element</p>
</div>

<style>
.clearfix:after {
    content: "";
    display: table;
    clear: both;
}
</style>
```

ALLES OVER CLEARFIX, EN AL ZIJN VARIANTEN:
HTTPS://CSS-TRICKS.COM/SNIPPETS/CSS/CLEAR-FIX/

FLOAT-BASED LAYOUTS:

Tot de komst van Flexbox & CSS Grid was dit de gangbare manier om een layout te maken. (Periode 2000-2016)
Met deze methode zijn Frameworks ontwikkeld om snel kolommen layouts te maken, zoals Bootstrap 3.

VOORBEELD:

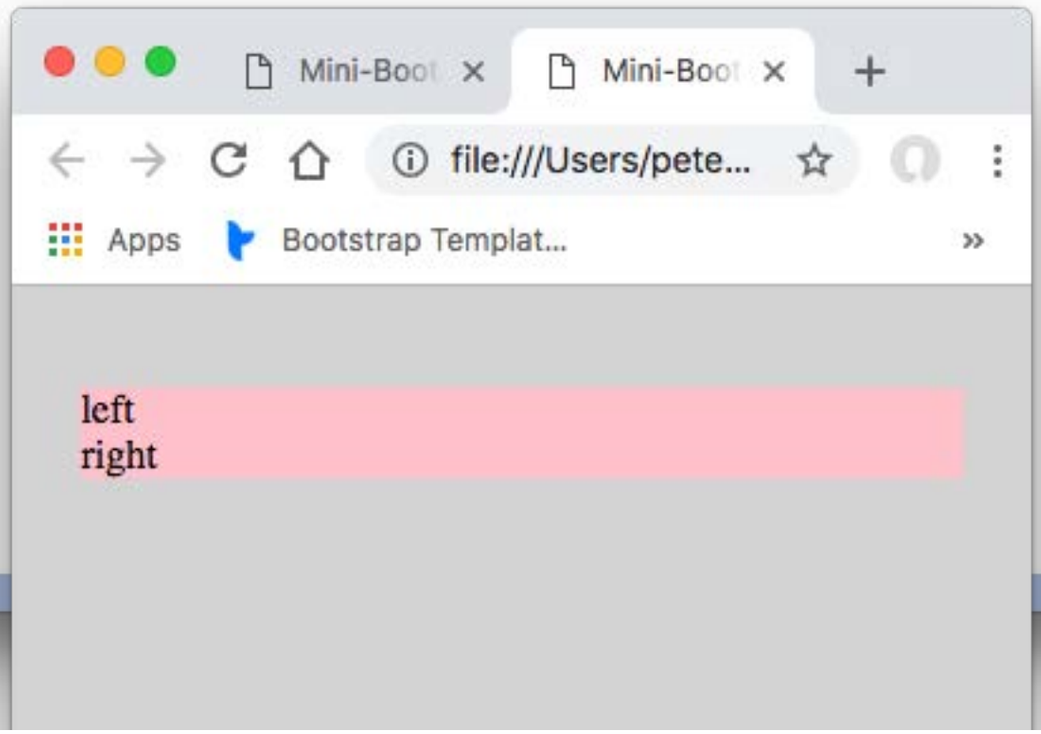
HTTPS://CODEPEN.IO/PETERVANDENHEUVEL/PEN/PXPGXP

```
<style>
.col-lg-11,
.col-xs-12,
.col-sm-12,
.col-md-12,
.col-lg-12 {
  position: relative;
  min-height: 1px;
  padding-right: 15px;
  padding-left: 15px;
}
.col-xs-1,
.col-xs-2,
.col-xs-3,
.col-xs-4,
.col-xs-5,
.col-xs-6,
.col-xs-7,
.col-xs-8,
.col-xs-9,
.col-xs-10,
.col-xs-11,
.col-xs-12 {
  float: left;
}
.col-xs-12 {
  width: 100%;
}
.col-xs-11 {
  width: 91.66666667%;
}
.col-xs-10 {
  width: 83.33333333%;
}
.col-xs-9 {
  width: 75%;
}



.col-xs-pull-12 {
  right: 100%;
}
.col-xs-pull-11 {
  right: 91.66666667%;
}
```

bootstrap-test.css

bootstrap-test.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Mini-Bootstrap 3</title>
    <link rel="stylesheet" href="bootstrap-test.css" />
</head>
<body>
    <div class="container">
        <div class="row clearfix">
            <div class="col-md-6 col-xs-12">left</div>
            <div class="col-md-6 col-xs-12">right</div>
        </div>
    </div>
</body>
</html>
```



bootstrap-test.css

```css
body { background-color: lightgrey;}

.container {
    width: 90vw;
    max-width: 1200px;
    margin: 0 auto;
    margin-top: 40px;
    background-color: white;
}

.clearfix:before,
.clearfix:after {
    content: " ";
    display: table;
}
.clearfix:after {
    clear: both;
}

.col-md-6,
.col-xs-12 { float: left; } /* Force columns next to another */

/* SMALL */
@media(min-width: 0px) and (max-width: 767px)  {
    .container{width: 90%;}
    .col-xs-12 {
        width: 100%;
        background-color: pink;
    }
}

/* Large */
@media(min-width: 768px) {
    .col-md-6 {
        width: 50%;
        background-color: yellow;
    }
}
```