CSS Repetition:¨
HTML minimale valide Variante

Minimale valide Variante

```
<!DOCTYPE html>
<title>Test</title>

<h1>Hello World</h1>

<p>Lorem ipsum dolor sit amet, consetetur</p>
```

unic

Neue Elemente: Auswahl

- Sections:
  `<section>`, `<nav>`, `<article>`, `<aside>`, `<header>`, `<footer>`, `<main>`
- Text-level semantics:
  `<time>`
- Embedded content:
  `<video>`, `<audio>`, `<picture>` **(5.1)**
- Forms:
  `<datalist>`, `<progress>`
- Interactive elements:
  `<details>` **(5.1)**

Übersicht: https://developer.mozilla.org/en/docs/Web/Guide/HTML/HTML5/HTML5_element_list


```
<head>
  <link rel="stylesheet" href="styles.css">
</head>
```

Block-level Elements

A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).

The element is a block-level element.

Example

```
<div>Hello</div>
<div>World</div>
```

Try it Yourself »

Block level elements in HTML:

<address>	<article>	<aside>	<blockquote>	<canvas>	<dd>		<dl>
<dt>	<fieldset>	<figcaption>	<figure>	<footer>	<form>	<h1>-<h6>	<header>
<hr>		<main>	<nav>	<noscript>		<output>	<p>
<pre>	<section>	<table>	<tfoot>		<video>		

Inline Elements

An inline element does not start on a new line and only takes up as much width as necessary.

This is an inline element inside a paragraph.

Example

```
<span>Hello</span>
<span>World</span>
```

Try it Yourself »

Inline elements in HTML:

<a>	<abbr>	<acronym>	<b>	<bdo>	<big>	 	<button>
<cite>	<code>	<dfn>		<i>	<img>	<input>	<kbd>
<label>	<map>	<object>	<q>	<samp>	<script>	<select>	<small>
		<sub>	<sup>	<textarea>	<time>	<tt>	<var>

The Element

The element is often used as a container for other HTML elements.

The element has no required attributes, but both **style** and **class** are common.

When used together with CSS, the element can be used to style blocks of content:

position: absolute
Ausrichtung ans Parent-Element: Main und Sidebar müssen deshalb left als Parameter setzen, da alle 3 Elemente sonst linksbündig wären.

1. "position: absolute": Probleme

- Elemente **ausserhalb des "Dokument-Flusses"**
  - Container wird nicht aufgezogen
  - nachfolgende Elemente werden nicht verdrängt
  - → **bei responsiven Webseiten gravierend**

- Lösungsansätze:
  - höchste Spalte statisch/relativ positionieren
    → Welche ist die höchste?
  - notwendige Höhe des Container mit JavaScript bestimmen
    → Zu welchem Zeitpunkt? Performance?

1. "position: absolute": 1. Versuch

- http://codepen.io/backflip/pen/xovfs

```
.nav,
.main,
.sidebar {
  position: absolute;
}
.main {
  left: 20%;
}
.sidebar {
  left: 80%;
}
```

Display: inline-Block (da gibts viele Probleme)

Ein Problem is mit inline-block der white-Space (z.B. Zeilenumbrüche oder einfach abstände im html! ächz): Wenn man die Abstände komplett entfernt oder mit Kommentar unnötig macht
dann wird das Alignment schon besser. HTML:

```
<body>
   <header class="header">header</header>

    <div class="content"><!--

        --><nav class="nav">nav</nav><!--
        --><div class="main">main<br>adsf</div><!--
        --><div class="sidebar">sidebarsd viel text. .. viel
text mach margin hack
```

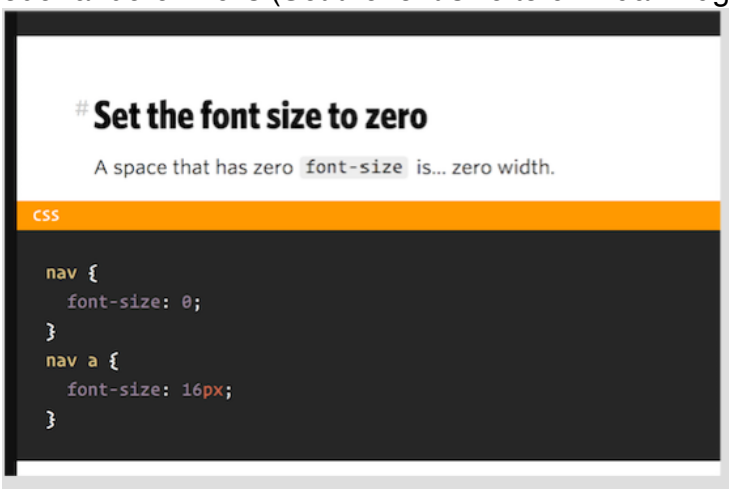
```
mit nur 10em zum problem -> 1000em oder mehr</div><!--
--></div>

   <footer class="footer">foot</footer>

</body>
```

Das CSS sieht dann so aus:
border-box (damit Padding korrekt dazugerechnet wird):
display: inlineBlock welches Element als ein inline-element rendered
vertical-align: top -> weil sonst die Elemente vom Elternelement (content) Baseline (bottom) ausgerichtet werden
margin-right: 0.27em => nötig um Whitespace (entspricht 0.27em - aber nur wenn fortgösse gleich gross bleibt!)loszuwerden (oder eben im HTML lösen, dass keinerlei whitespace gibt)

oder andere Tricks (Set the font size to 0 ! wuah - ugly):



margin-bottom: -10em; (10em ist übrigens zuwenig - der Platz reicht nicht aus bei grossen Inhalten)
padding-bottom: 11em:
-10 lässt das Element verschwinden während padding-bottom das wieder aufzieht - aber eben relevant als box-sizing: border-box, welche das Padding mitrechnet.

```
.nav,
.main,
.sidebar {
    box-sizing:border-box;
    display: inline-block;
    /*margin-right:-0.27em;*/
    vertical-align: top;

    margin-bottom: -10em;
    padding-bottom: 11em;

}
```

```
.content {
    overflow: hidden;
}
```

overflow: hidden ist nötig, damit überlappende Block-Element nachfolgendes Element nicht verdecken - wegen dem padding-bottom: 11em Hack - dieser ist sonst nicht nötig.

Weiter Möglichkeiten::
Man könnte auch Hintergrundbilder reintun, respektive mehrere Hintergrundbilder, oder auch gradients
oder eben wir im css verwendet - margin-bottom und padding-bottom. Oder per Javascript Height setzen

Float:left
Hier braucht auch wieder der hack mit margin und padding um .nav und .sidebar aufzuziehen.
overflow: hidden damit nicht footer und alles danach verdeckt wird.

```
.nav,
.main,
.sidebar {
    float: left;
    margin-bottom: -10em;
    padding-bottom: 11em;
}

.content {
    overflow: hidden;
}
```

display: Table-Cell:
display: Table-cell zieht container nicht ganz auf. Es braucht auf dem eltern-element (.content) auch noch display: table und width = 100%; sonst wird er nur bis auf ca. 40 - 60% des containers aufgezogen

```
.nav,
.main,
.sidebar {
    display: table-cell;
}

.content {
```

```
    display: table;
    width: 100%;
}
```

display: flex (er teilt sich schön auf)

```
.content {
    display: flex;
}
```

display: grid (Grid ist am umfangreichsten und kann verschiedentlich konfiguriert werden (und auch sehr kompliziert)

```
.content {
    display: grid;
    grid-template-columns: 20% auto 20%;
}
```

Zusammenfassung:

Zusammenfassung

- Bisher meistverbreitete Lösung: **Floats**
  - Gleich hohe Spalten nur mit Hacks möglich
  - Standard bei den meisten Frameworks
- Neue Standardlösung: **Flexbox**
  - Allenfalls Fallback für ältere Browser notwendig
- Vermutlich zukünftige Standardlösung für komplexere Layouts: **Grid Layout**
  - http://www.w3.org/TR/css-grid-1
  - Beispiel: http://www.w3.org/TR/css-grid-1/#source-independence