nth-of-type (nth Element of type p von seinen Eltern)

```
<!DOCTYPE html>
<html>
<head>
<style>
p:nth-of-type(2) {
    background: red;
}
</style>
</head>
<body>

<p>The first paragraph.</p>
<p>The second paragraph.</p>
<p>The third paragraph.</p>
<p>The fourth paragraph.</p>

</body>
</html>
```

The first paragraph.

The second paragraph.

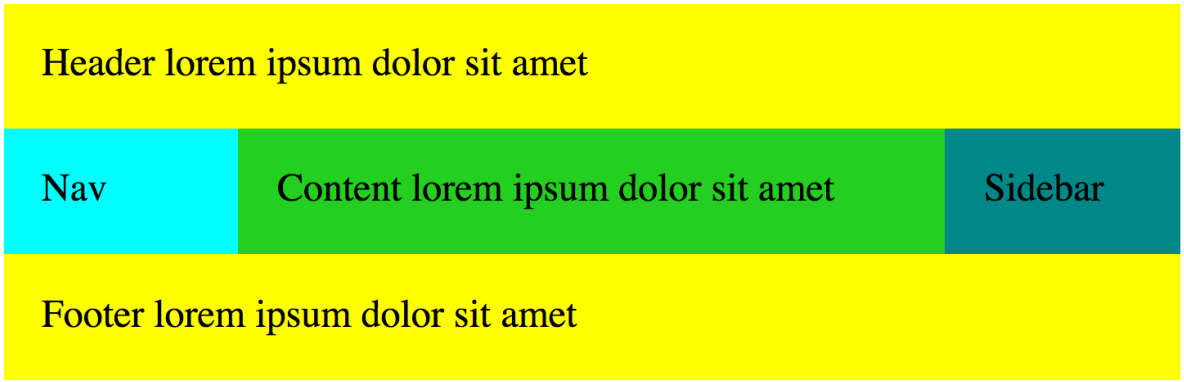
The third paragraph.

The fourth paragraph.

css_repetition

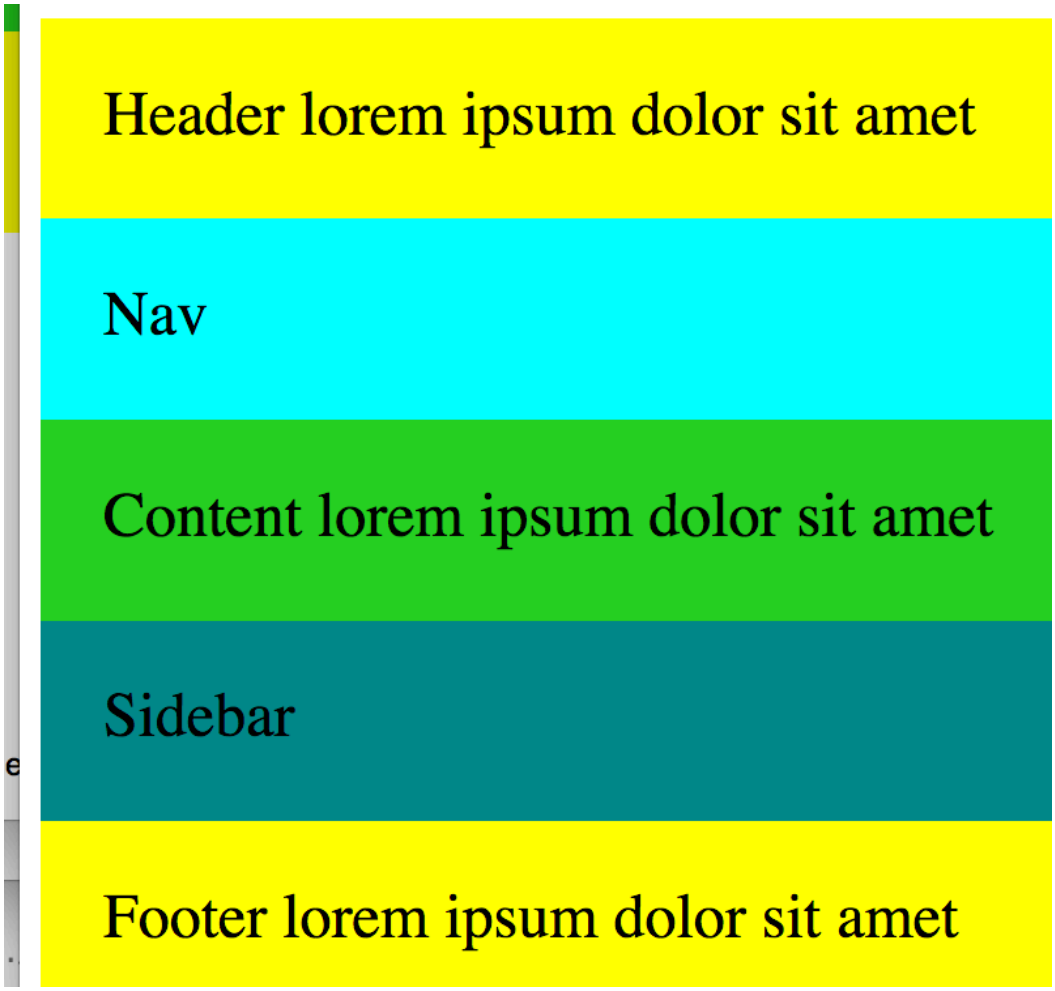
responsive und Fluid

Wieder vorherige Beispiel:




```
/* Responsive */

@media (max-width: 30em) {
  .content {
    flex-direction: column;
  }
  .nav,
  .main,
  .sidebar {
    width: auto;
  }
}
```

so kommt nachher alles untereinander (und width:auto macht die Spaltenbreite wieder auto-berechnet):



Responsive Device Grössern 2014
deshalb responsive Design
Fluids Design (arbeitet „nur“ mit % aber Elemente bleiben gleich angeordnet)
früher gabs Flashseiten(2004)
2010 -> Responsive Design (rdp)

Media queries:

Media Queries: Operatoren/Keywords

```
AND: @media (min-width: 20em) and (max-width: 30em)

OR:  @media (max-width: 10em),
       (min-width: 20em) and (max-width: 30em),
       (min-width: 40em) {}

NOT: @media not screen {}
     @media not screen and (min-width: 20em) {}
     @media not screen and (min-width: 20em),
       (min-width: 40em) {}


ONLY (“Schutz” vor alten Browsern):
     @media onlv screen and (min-width: 20em) {}
```

Mediaqueries mit % macht keinen Sinn (% of what?)

- px sind "CSS-Pixel"
  - Abstraktion für uns Entwickler
  - Verhältnis *CSS-Pixel / echte Pixel* wird durch Zooming verändert
  - Initiales Verhältnis wird (willkürlich) durch den Browser gesetzt

Einschub: Accessibility

- Anti-Patterns, verhindern pinch-to-zoom (neue Browser ignorieren es zum Teil aus diesem Grund):

```
<meta name="viewport" content="width=device-width, initial-
scale=1.0, user-scalable=no" />

<meta name="viewport" content="width=device-width, initial-
scale=1.0, maximum-scale=1" />
```

Zirkuläre Abhängigkeiten vermeiden mit containment (um Elemente unabhängig zu machen vom rest)

The **`contain`** property allows an author to indicate that an element and its contents are, as much as possible, *independent* of the rest of the document tree. This allows the browser to recalculate layout, style, paint, size, or any combination of them for a limited area of the DOM and not the entire page.

```
/* No layout containment. */
contain: none;
```

MENU und REspnsives Verhalten:
ein extra menu für Sidebar machen im Html und mit Display none / visible entsprechend setzen (klasse .open verwenden)

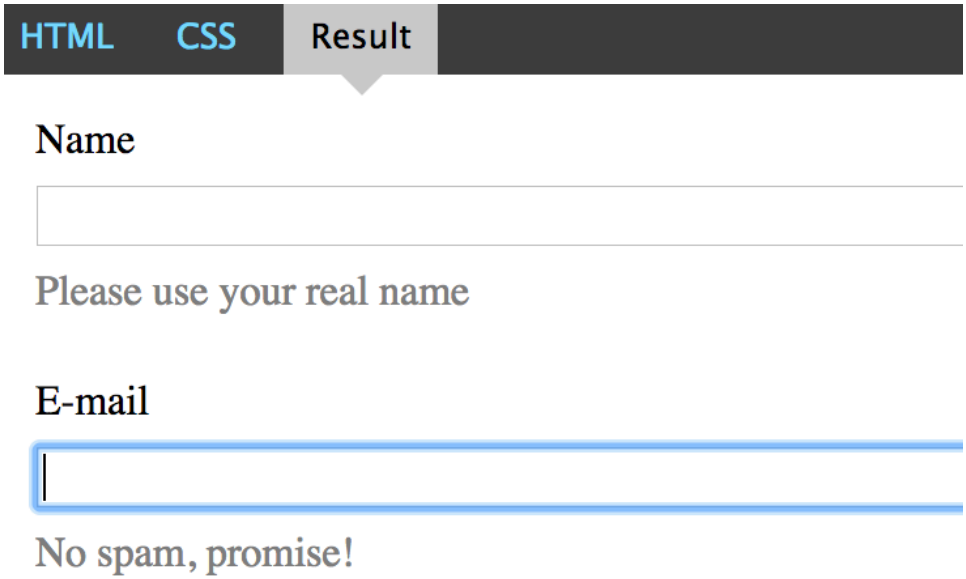
```
<div class="nav">
  <span class="nav-item">Services</span>
  <span class="nav-item">About us</span>
  <span class="nav-item">Contact</span>
</div>
<div class="menu-button">
  Menu
</div>

<div class="side-menu">
  <h3>Services</h3>
  <span class="sub-nav-item">Webdesign</span>
  <span class="sub-nav-item">Usability</span>
  <span class="sub-nav-item">Accessibiity</span>
  <h3>About us</h3>
  <span class="sub-nav-item">Company</span>
  <span class="sub-nav-item">Awards</span>
  <h3>Contact</h3>

</div>
```

```scss
.side-menu {
  display: none;
  padding: 50px 20px 20px 20px;
  height: 100vh;
  width: 200px;
  border-left: 1px solid grey;

  h3 {
    margin-bottom: 5px;
  }

  &.open {
    display: flex;
    flex-direction: column;
  }

  .sub-nav-item {
    position: relative;
    left: 20px;
  }
}

@media (min-width: 750px) {
  .side-menu.open {
    display: none;
  }
}

@media (max-width: 750px) {
  .menu-button {
    display: flex;
  }

  .nav-item {
    display: none;
  }

  .nav {
    height: 19px;
  }
}
```

```
<script>
$('.menu-button').click((event) => {
  $('.menu-button').toggleClass('open');
  $('.side-menu').toggleClass('open');
```

})</script>

Nochmals Tabes und Media Queries:



HTML CSS Result
Edit on

```
<form action="#">
  <div class="row">
    <label for="name">Name</label>
    <input type="text" id="name" name="name" aria-
describedby="hint-name" class="field">
    <span class="hint" id="hint-name">Please use your
real name</span>
  </div>
  <div class="row">
    <label for="email">E-mail</label>
    <input type="email" id="email" name="email" aria-
describedby="hint-email" class="field">
    <span class="hint" id="hint-email">No spam,
promise!</span>
  </div>
</form>
```

HTML CSS Result
Edit on

```css
body {
  margin: 1em;
}

.row {
  display: flex;
  align-items: center;
  margin-bottom: 0.5em;
}

label {
  flex: 1;
  padding-right: 1em;
  text-align: right;
}
.field {
  box-sizing: border-box;
  font-size: 1em;
  flex: 2;
}
.hint {
  color: gray;
  flex: 2;
  padding-left: 1em;
}

@media (max-width: 30em) {
  .row {
    flex-direction: column;
    align-items: flex-start;
    margin-bottom: 1.5em;
```

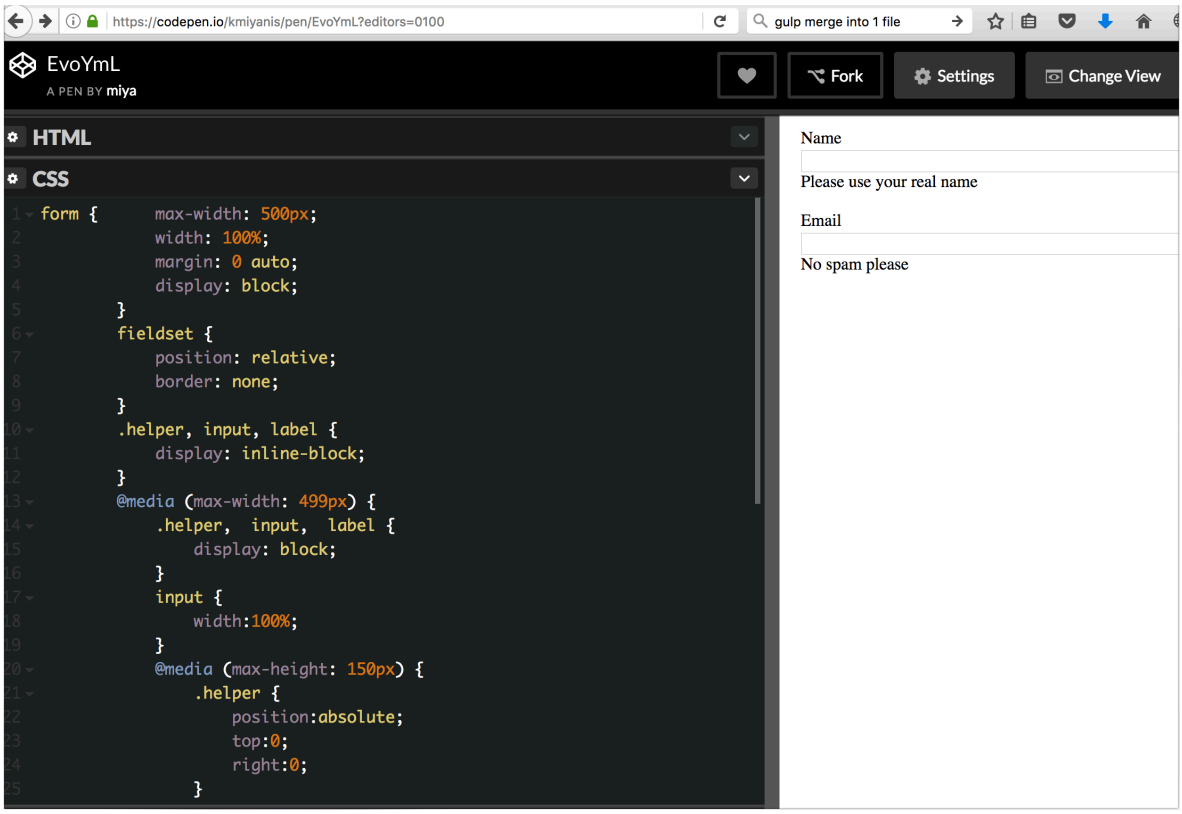
```
  }

  label {
    margin-bottom: 0.5em;
    padding-right: 0;
    text-align: left;
  }
  .field {
    width: 100%;
  }
  .hint {
    margin-top: 0.5em;
    padding-left: 0;
  }
}

@media (max-width: 30em) and (max-height: 15em) {
  .row {
    flex-direction: row;
    flex-wrap: wrap;
  }

  .field {
    order: 2;
    flex: 100%;
  }
  .hint {
    margin-top: 0;
    margin-bottom: 0.5em;
    text-align: right;
  }
}
```

Beispiel von Miya:
Setzen von verschachtelten Media Queries und anpassen :

HTML:

```
<form>
   <fieldset>
      <label for="name">Name</label>
      <input type="text" id="name">
      <div class="helper">Please use your real name</div>
   </fieldset>
   <fieldset>
      <label for="email">Email</label>
      <input type="email" id="email">
      <div class="helper">No spam please</div>
   </fieldset>
</form>
```

CSS:



Images: und optimales image:

```
<img srcset="large.jpg  1024w,
             medium.jpg 640w,
             small.jpg  320w"
     sizes="(min-width: 64em) 25vw,
            100vw"
     src="small.jpg"
     alt="Awesome photo" />
```

64em mind.breite des Screens. dann wird ein Bild 25% des bildschirmes angezeigt, wenn weniger als 64em dann 100% der Bildschri.Breite.
Im 1. Fall (25%) sucht sich nacher der Browser das der bildschirm.breite
entsprechende Bild (wenn bildschirm also 1600w (px?) / 24% = 400w (=400px?) dann holt sich browser 640w-Bild

wenn mindestens 64em dann soll er 25vw anwenden (25% der Bildschirmes - und das kann der Browser ausrechnen) ur so kann browser entscheiden was zu laden ist

Use Case 3: Art Direction

- Unterschiedliche **Bildausschnitte** je nach Screengrösse:

```
<picture>
   <source media="(min-width: 64em)" srcset="portrait.jpg">
   <source srcset="landscape.jpg">

   <img src="landscape.jpg" alt="Awesome photo">
</picture>
```

picture: browser muss sich daran halten , auche wenn er wegen bandbreite ev. was anderes laden möchten

How to use the <picture> tag:

```
<picture>
  <source media="(min-width: 650px)" srcset="img_pink_flowers.jpg">
  <source media="(min-width: 465px)" srcset="img_white_flower.jpg">
  <img src="img_orange_flowers.jpg" alt="Flowers" style="width:auto;">
</picture>
```

Example

```
/* For width smaller than 400px: */
body {
    background-image: url('img_smallflower.jpg');
}

/* For width 400px and larger: */
@media only screen and (min-width: 400px) {
    body {
        background-image: url('img_flowers.jpg');
    }
}
```

min-width is browserbreite: man kann auch device-width (Gerätebreite - bleibt gleich beim resizen des Browser):

You can use the media query `min-device-width`, instead of `min-width`, which checks the device width, instead of the browser width. Then the image will not change when you resize the browser window:

Example

```
/* For devices smaller than 400px: */
body {
    background-image: url('img_smallflower.jpg');
}

/* For devices 400px and larger: */
@media only screen and (min-device-width: 400px) {
    body {
        background-image: url('img_flowers.jpg');
    }
}
```

Try it Yourself »

DESIGN FRONTEN : AUswirkungen:

Ausgangslage

- **Designprozess ist anspruchsvoller**, man muss z.T. stark unterschiedliche Zustände beachten.
  - Verschiedene Devices, Screengrössen, Eingabemedien.
  - Gewichtung und Priorisierung von Elementen je nachdem anders
- **Kommunikation** (mit uns Entwicklern, mit dem Kunden, …) und Dokumentation **ist anspruchsvoller**:
  - Demonstration von Design und Interaktionen
  - Übergabe von Artefakten
  - Dokumentation