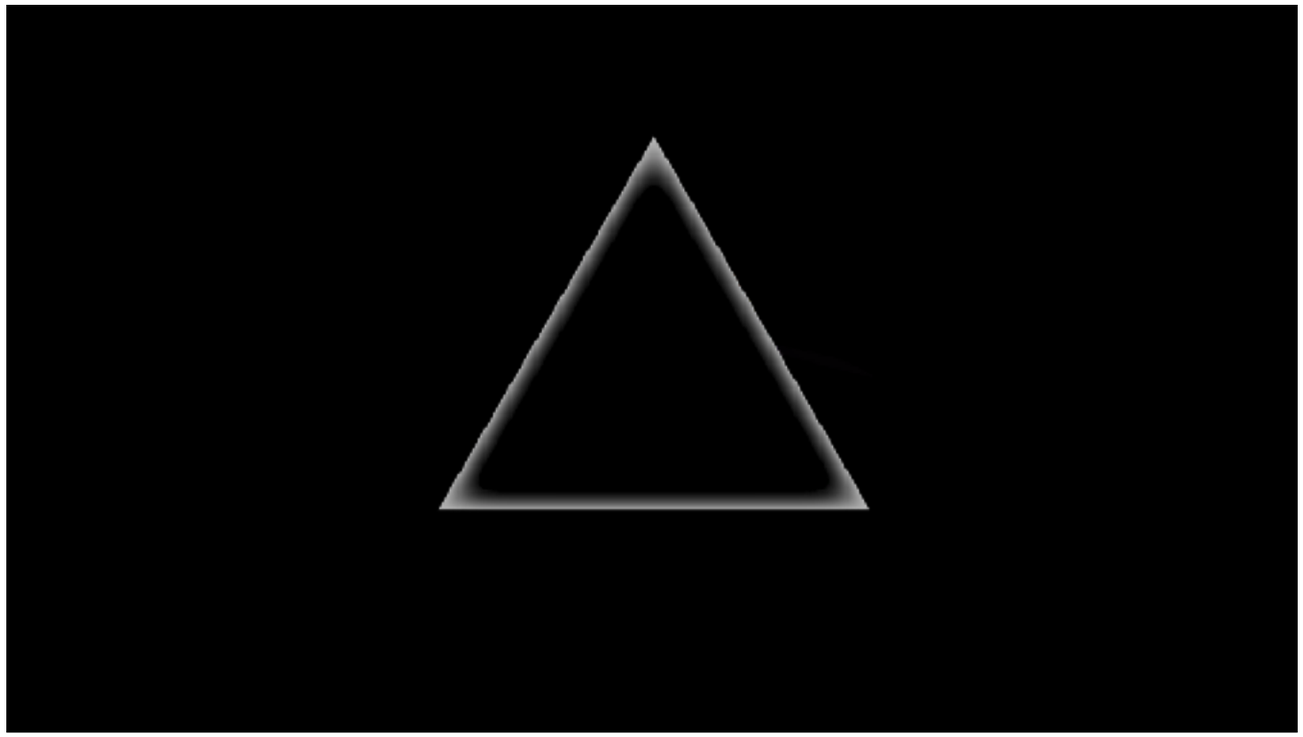


Manuel Matuzovic

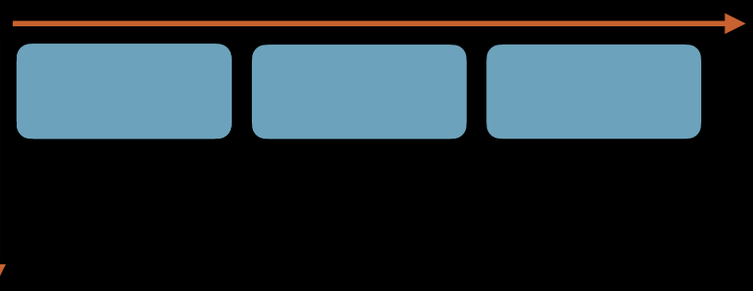
- Frontend Developer aus Wien
- Spezialisiert auf CSS, Performance und Accessibility
- Schreibe Artikel, halte Vorträge und Workshops
- webclerks und CodePen Meetup in Wien
- @mmatuzo auf Twitter

- Ich bin Frontend Developer aus Wien
- Bin spezialisiert auf CSS, Performance und Accessibility
- Schreibe Artikel, halte Vorträge und Workshops über diese Themen
- Ich organisiere das webclerks und CodePen Meetup in Wien
- Auf Twitter bin ich unter @mmatuzo zu finden.

- Heute bin ich hier, um über CSS Grid Layout zu sprechen
- Einerseits möchte ich zeigen, wie umfangreich und welch ein Segen es ist, aber ich möchte auch die dunkle Seite zeigen.
- Zu Beginn werde ich kurz erklären, was Grid ist und die Geschichte sehr grob abbilden.

Was ist CSS Grid Layout?

CSS Grid ist ein auf Rastern basierendes Layout-System, das für zweidimensionale Layouts designt wurde.



@mmatuzo

- Grid ist ein Layout-System für zweidimensionale Layouts.
- Es eignet sich perfekt für Layouts, die sich über zwei Ebenen erstrecken. Also, wenn man nicht nur auf der x oder y Achse arbeitet, sondern auf beiden gleichzeitig.

Was ist besonders an Grid?

- Das erste richtige Layout-System in CSS.
- **`float`**, **`display: inline-block`**, **`position`**, **`display: table`** waren ursprünglich nicht für das Bauen von Layouts konzipiert.
- Irgendwie wie table-Layouts, aber responsive, flexibel und in CSS formuliert und nicht in HTML.

@mmatuzo

- Das besondere an Grid ist, dass es das erste richtige Layout-System in CSS ist.
- Man fragt sich vielleicht, wie das sein kann. CSS gibt es doch schon seit 20 Jahren.
- Keine Ahnung. Wir hatten bis jetzt immer nur float, inline block und position zu Verfügung. Natürlich hat Flexbox einiges einfacher gemacht, aber Flexbox eignet sich nur wirklich, wenn man auf einer Achse arbeitet. Manche arme Menschen unter uns, mich eingeschlossen, mussten noch Layouts mit HTML Tables bauen.
- Interessant ist, dass Grid irgendwie wie arbeiten mit Tables ist, aber responsive, flexibel und in CSS geschrieben und nicht in HTML indem man Elemente hinzufügt und diese dann das Grid formen.

Grids Geschichte in Kürze

- 2011: Microsoft liefert die erste Implementierung von Grid hinter dem `-ms-` Browserprefix in IE 10.
- Jänner 2017: CSS Grid in Chromium 56 für Android
- Früher März 2017: Chrome und Firefox
- Ende März 2017: Opera und Safari
- 17. Oktober 2017: Edge

@mmatuzo

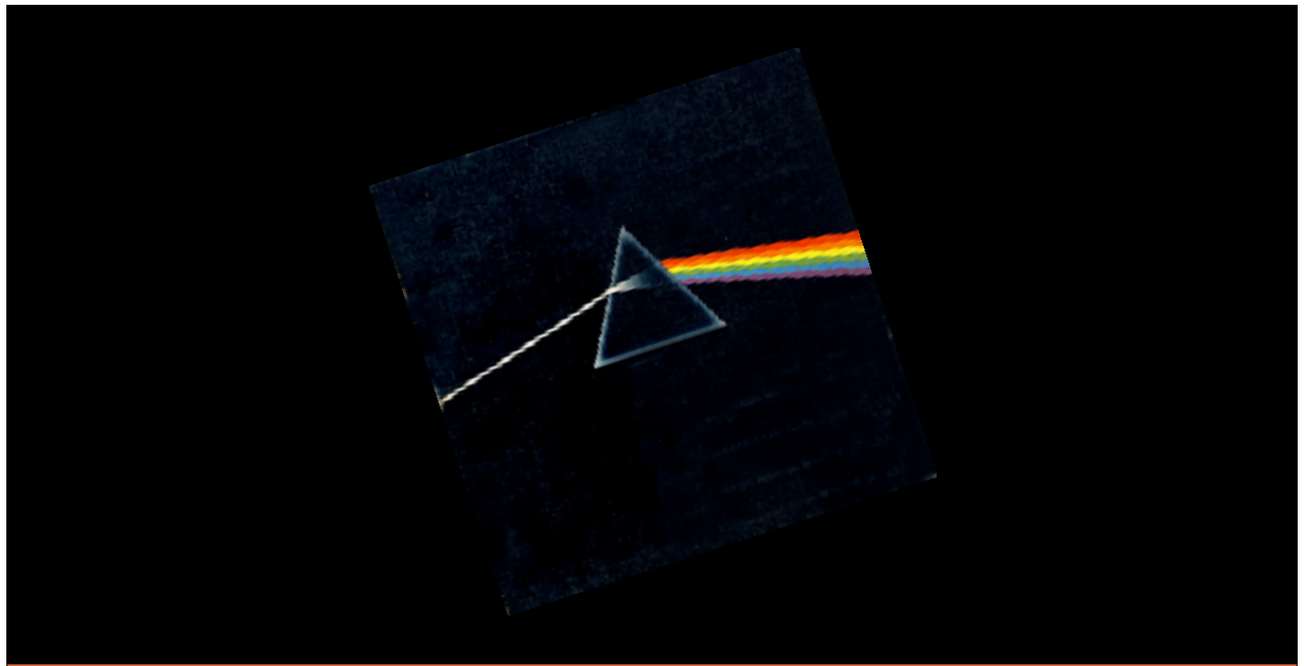
- Grid wurde von Microsoft erfunden, weil sie dringend eine derartige Layout-Technik für Windows 8 benötigten.
- Die erste Implementierung von Grid gab es bereits 2011 in IE10 hinter dem -ms- Prefix.
- Alle anderen Browser sind mit einer anderen, erweiterten Spezifikation 2017 gefolgt.
- Das ist schon etwas sehr besonderes, weil alle Browser diese umfangreiche Spezifikation innerhalb eines Jahres implementiert haben.
- Das gab es vorher in dem Umfang noch nicht.

Grid(-verwandte) Eigenschaften und Werte

- display
- grid-template-columns
- grid-template-rows
- grid-template-areas
- grid-template
- grid-column-gap
- grid-row-gap
- grid-gap
- justify-items
- align-items
- place-items

@mmatuzo

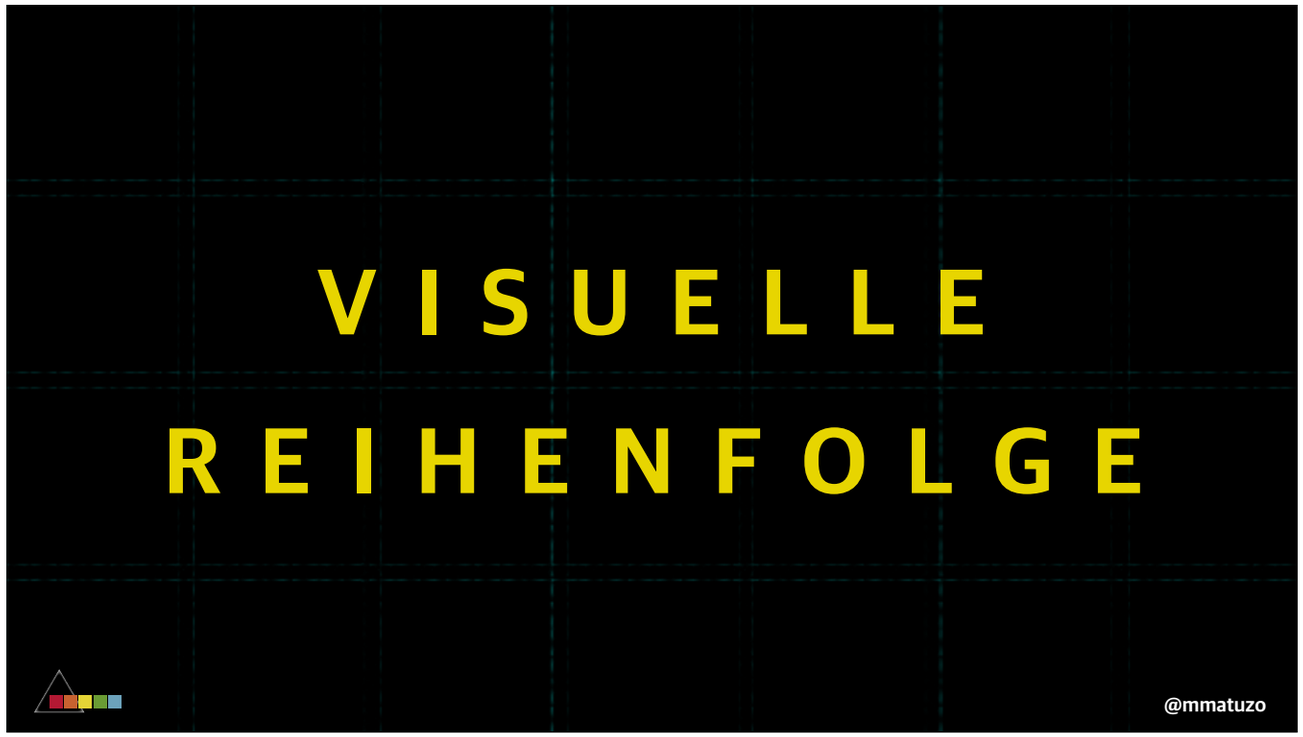
- Ich habe leider nicht genug Zeit, Grid genauer vorzustellen, aber ich kann nur sagen, dass es unglaublich umfangreich ist.
- Es gibt eine Vielzahl neuer Eigenschaften, Werte, Einheiten und verwandte Spezifikationen.
- Bevor ich zum Hauptteil komme, muss ich noch kurz eine Sache ansprechen.



Cover des Albums „The Dark Side of the Moon“, 1973 erschienen.

- Der Name des Vortrags sowie das Design basieren auf dem 1973 erschienenen Album „The Dark Side of the Moon“ von Pink Floyd.
- Ich bin ein riesen Pink Floyd, ich weiß alles über die Band. Nein, Blödsinn. Vor den Vorbereitungen wusste ich so gut wie nichts.
- Ich bin mir aber sicher, dass ich nicht der einzige bin, der so ungebildet ist.
- Deswegen möchte ich auch in diese Richtung bilden.

- The Dark Side of the Moon ist mit über 45 Millionen verkauften Tonträgern auf Platz 3 der meistverkauften Alben weltweit.
- Auf Platz zwei ist Back in Black von AC/DC mit 50 Millionen
- Auf Platz eins ist Thriller von Michael Jackson mit 66 Millionen



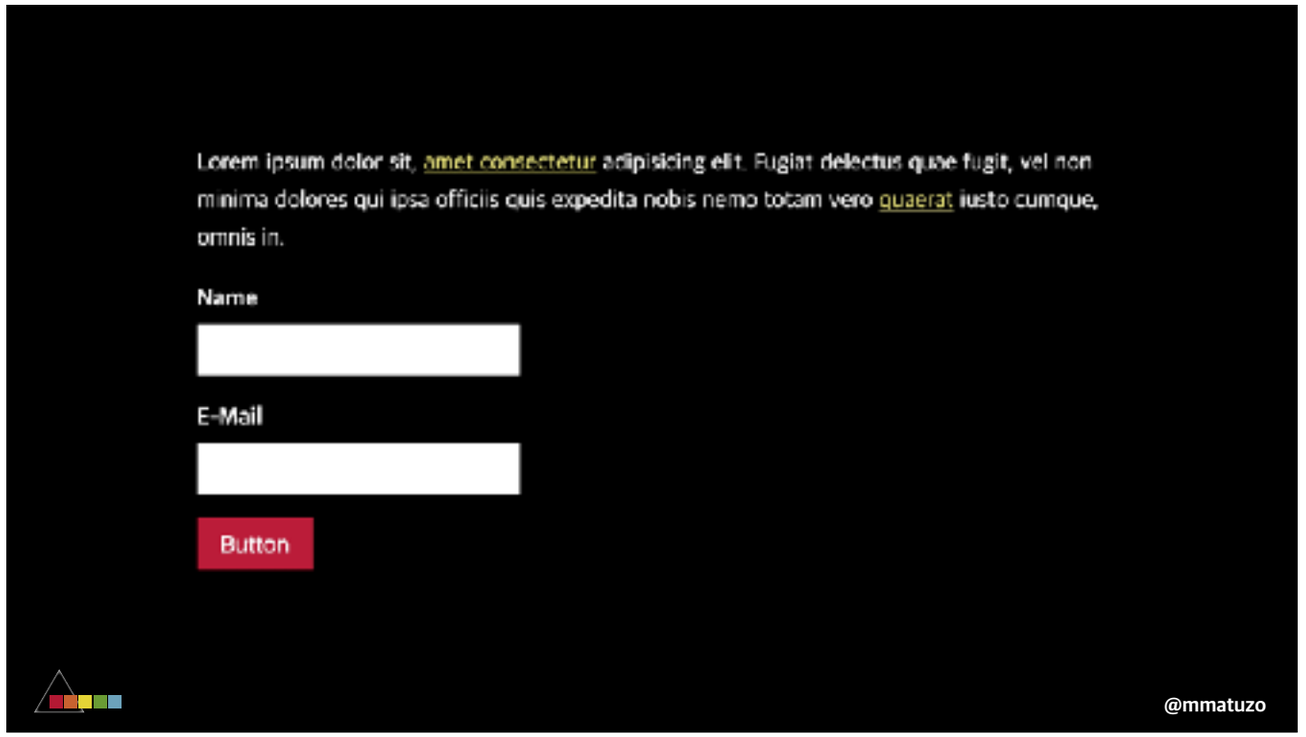
- Kommen wir nun zum Hauptteil
- Das erste und größte Thema ist visuelle Reihenfolge.
- Bevor ich auf Grid eingehe, möchte ich kurz die grundlegende Problematik erklären.

Visuelle Reihenfolge

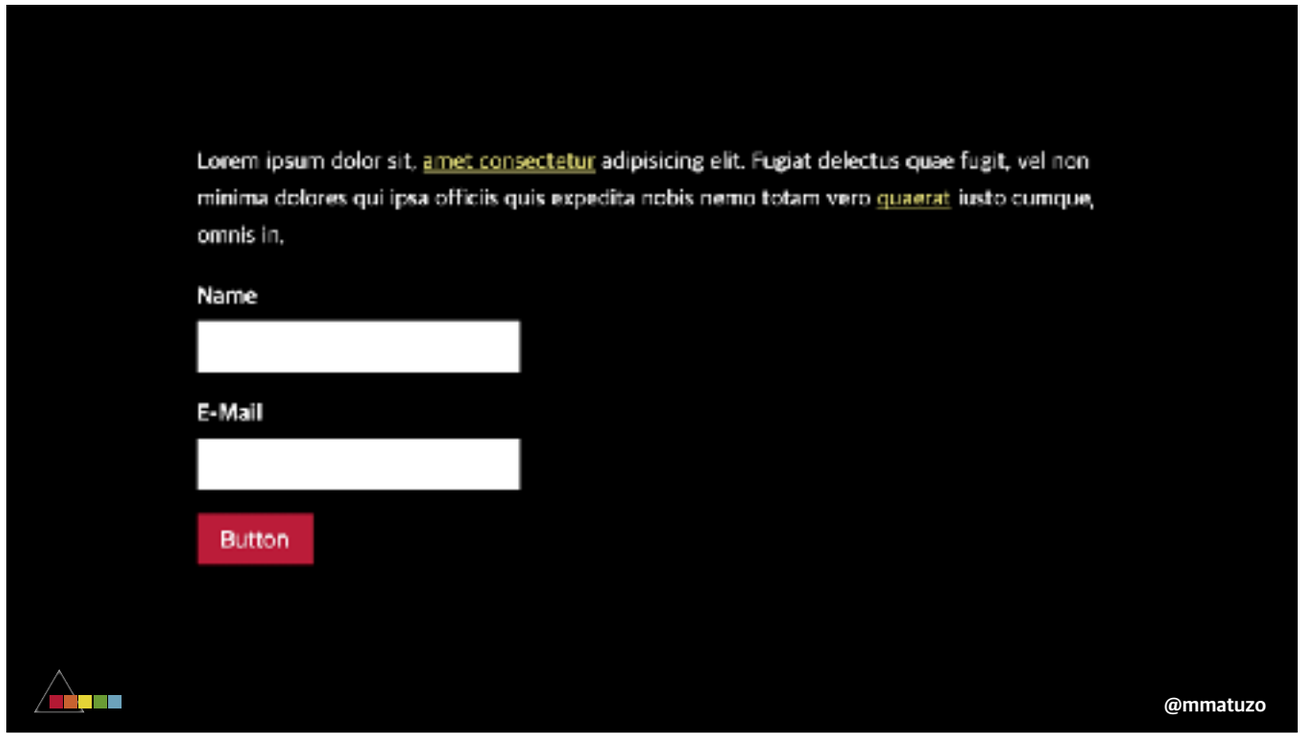
Sowohl die Tab-Reihenfolge als auch die Reihenfolge in der Screenreader Inhalte lesen folgen der Reihenfolge im DOM.

@mmatuzo

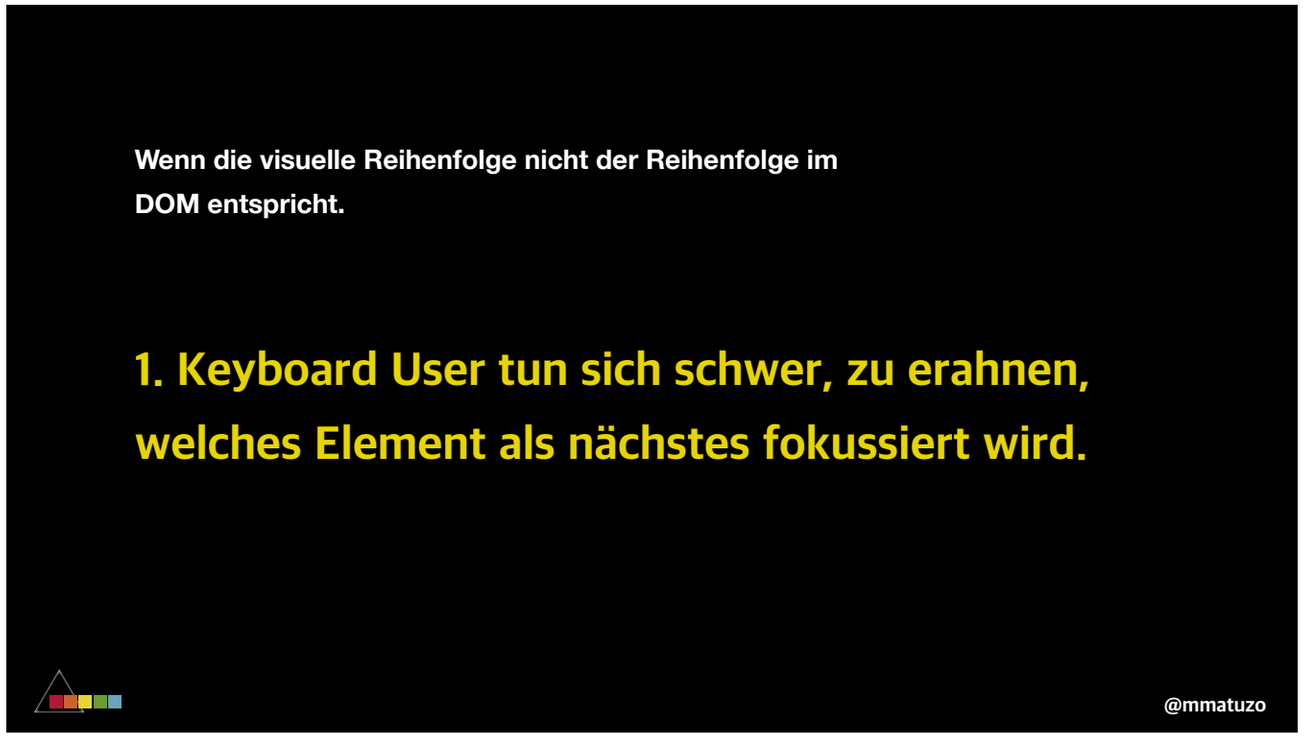
Sowohl die Tab-Reihenfolge als auch die Reihenfolge in der Screenreader Inhalte lesen folgen der Reihenfolge im DOM, also der Reihenfolge im HTML.



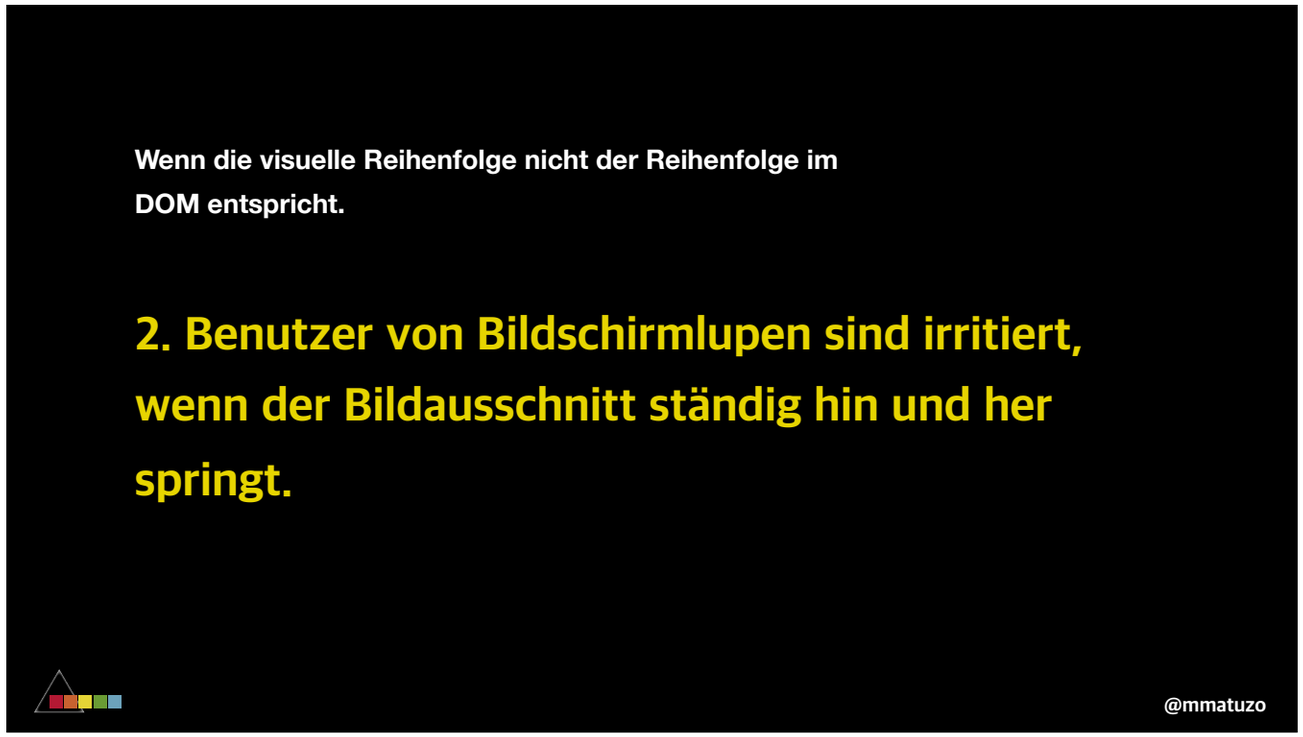
- Das heißt, wenn ich die Elemente durchtabbe, bewegt sich der Fokus von oben nach unten, von ersten zum letzten Element im HTML.



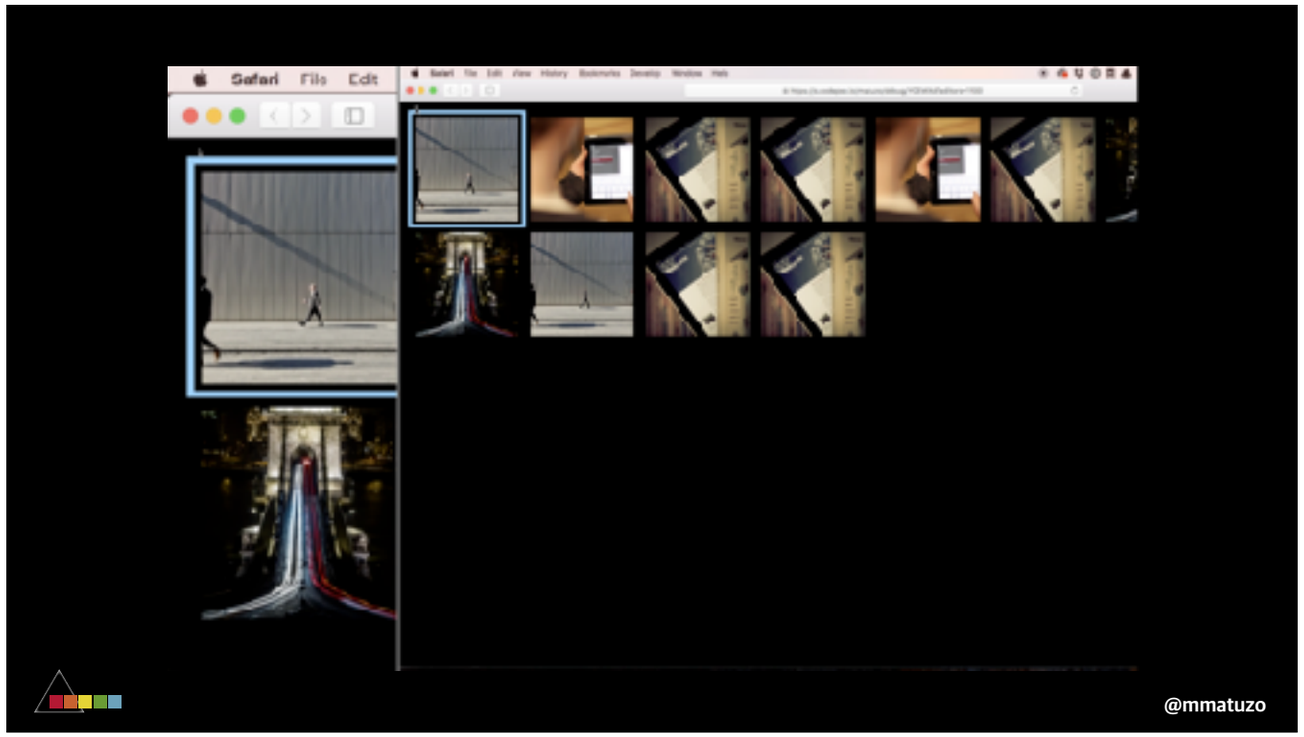
- Wenn man die visuelle Reihenfolge ändert, stellt das ein Problem da, weil die Tab und Screenreader-Reihenfolge immer noch der Reihenfolge im DOM (document object model) folgen.
- Man sieht hier, dass der Fokus jetzt wild hin und her springt.



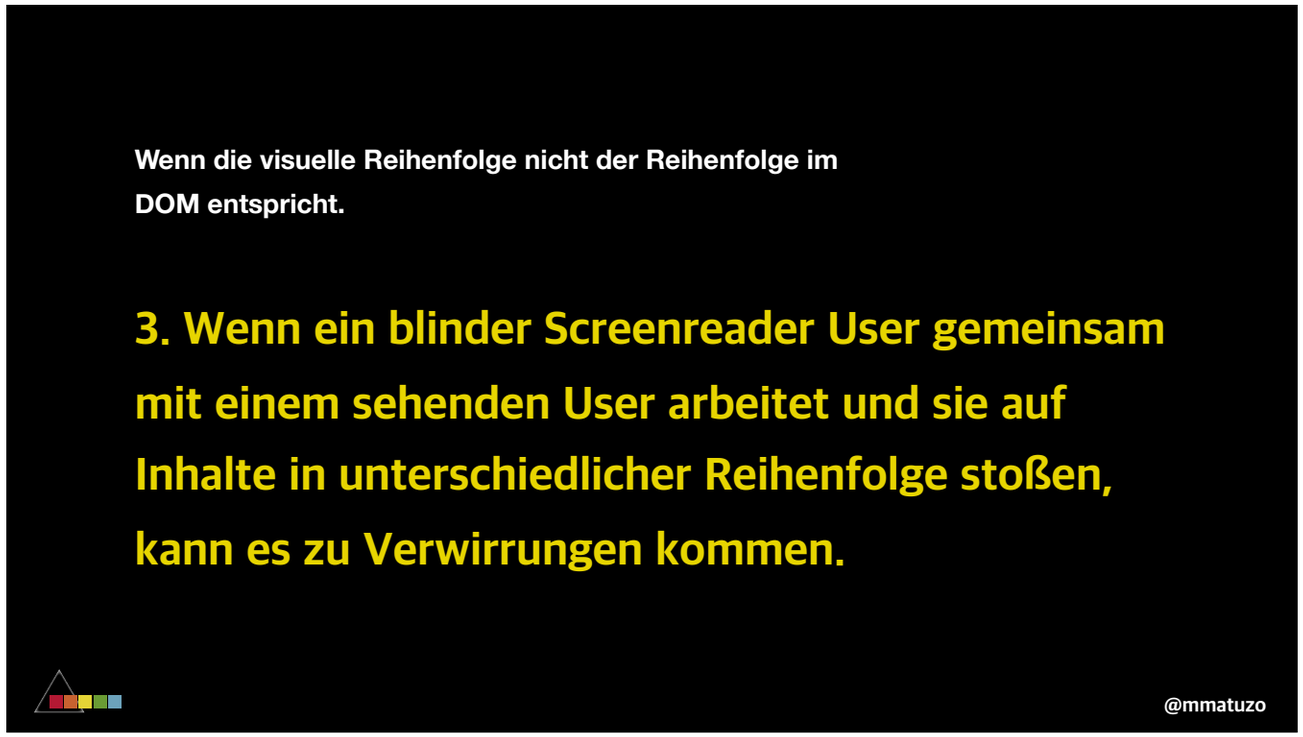
Wenn die visuelle Reihenfolge nicht der Reihenfolge im DOM entspricht, ist das schlecht weil Keyboard User sich schwer damit tun, zu erahnen, welches Element als nächstes fokussiert wird.



Weiters könnten Benutzer von Bildschirmlupen irritiert werden, wenn der Bildausschnitt ständig hin und her springt.



- Hier verwende ich die native Bildschirm Lupe in macOS und man sieht links den vergrößerten Ausschnitt.
- Ist natürlich nur ein einfaches Beispiel, aber es zeigt ganz gut wie unruhig die Experience wird, wenn die Reihenfolge nicht linear.



Außerdem kann es zu Verwirrungen kommen, wenn ein blinder Screenreader User gemeinsam mit einem sehenden User arbeitet und sie auf Inhalte in unterschiedlicher Reihenfolge stoßen.

Wer navigiert mit dem Keyboard?

- Menschen, die aus physischen Gründen keine Maus bedienen können.
- Menschen mit chronischen Erkrankungen, die körperliche Belastung meiden sollten.
- Jemand, der/die temporär keine Maus verwenden kann (bspw. wegen einer Verletzung)
- Poweruser

@mmatuzo

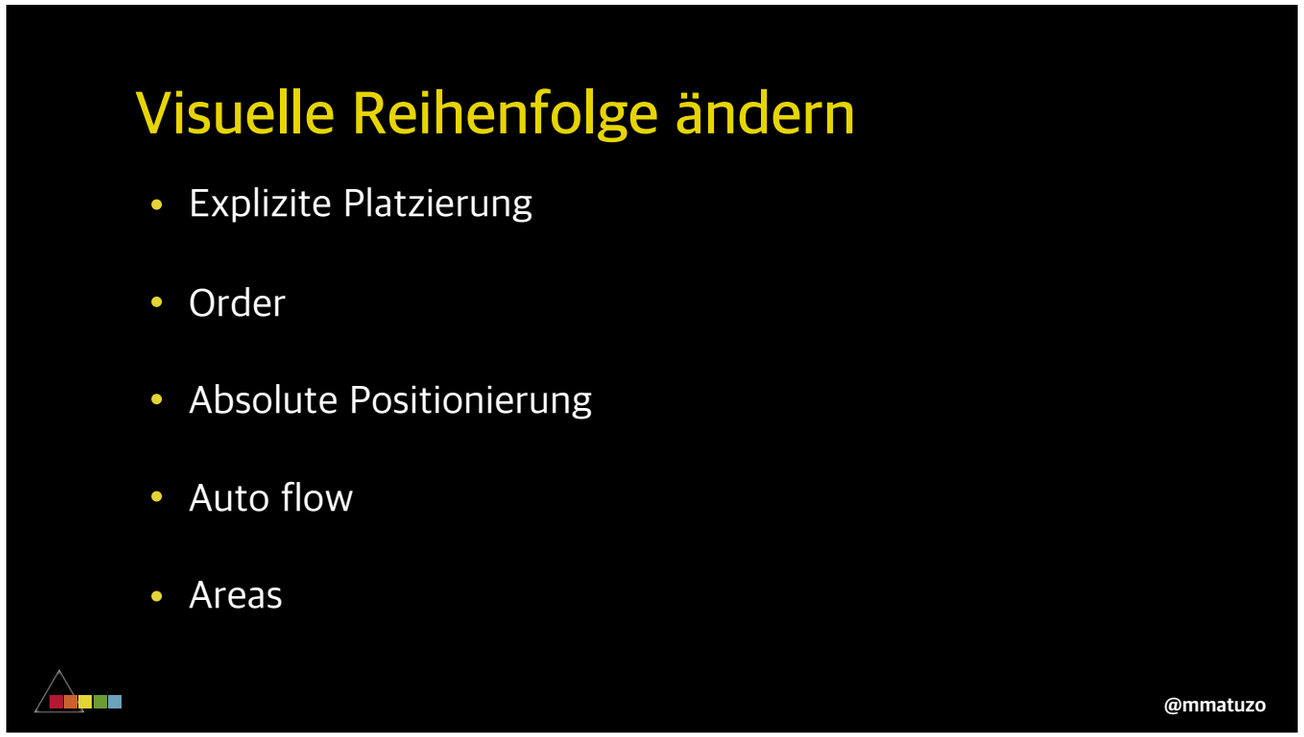
Wen betrifft das eigentlich? Wer navigiert mit dem Keyboard?
- Menschen, die aus physischen Gründen keine Maus bedienen können.
- Menschen mit chronischen Erkrankungen, die körperliche Belastung meiden sollten.
- Jemand, der/die temporär keine Maus verwenden kann (bspw. wegen einer Verletzung). Habe bei einer Konferenz eine Dame kennengelernt die 2 Wochen keine Maus bedienen konnte, weil sie den Unterarm frisch tätowiert hatte.
- Poweruser, also auch wir. Wir arbeiten sehr viel mit der Tastatur. Warum nicht auch im Browser?

Wer verwendet Screenreader?

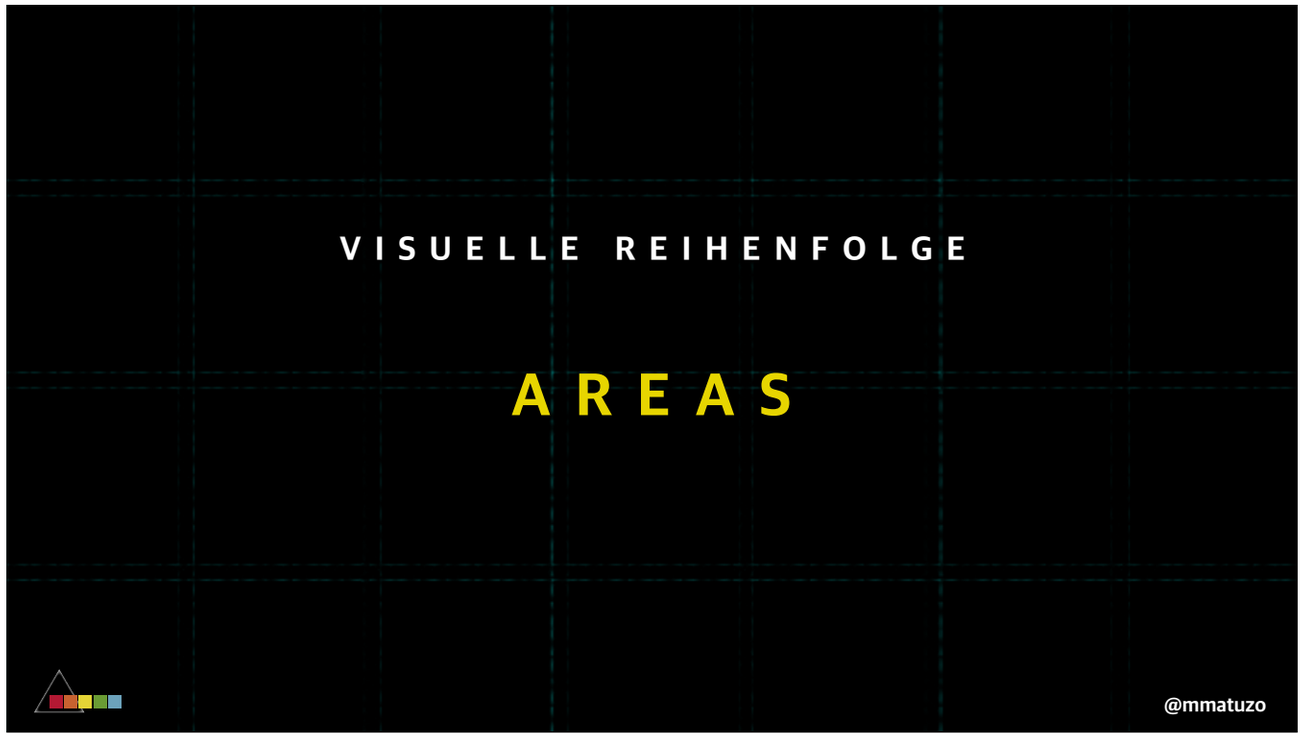
- Blinde Menschen.
- Menschen mit Sehschwächen, die ergänzend einen Screenreader verwenden.
- Menschen mit Lernschwächen.

@mmatuzo

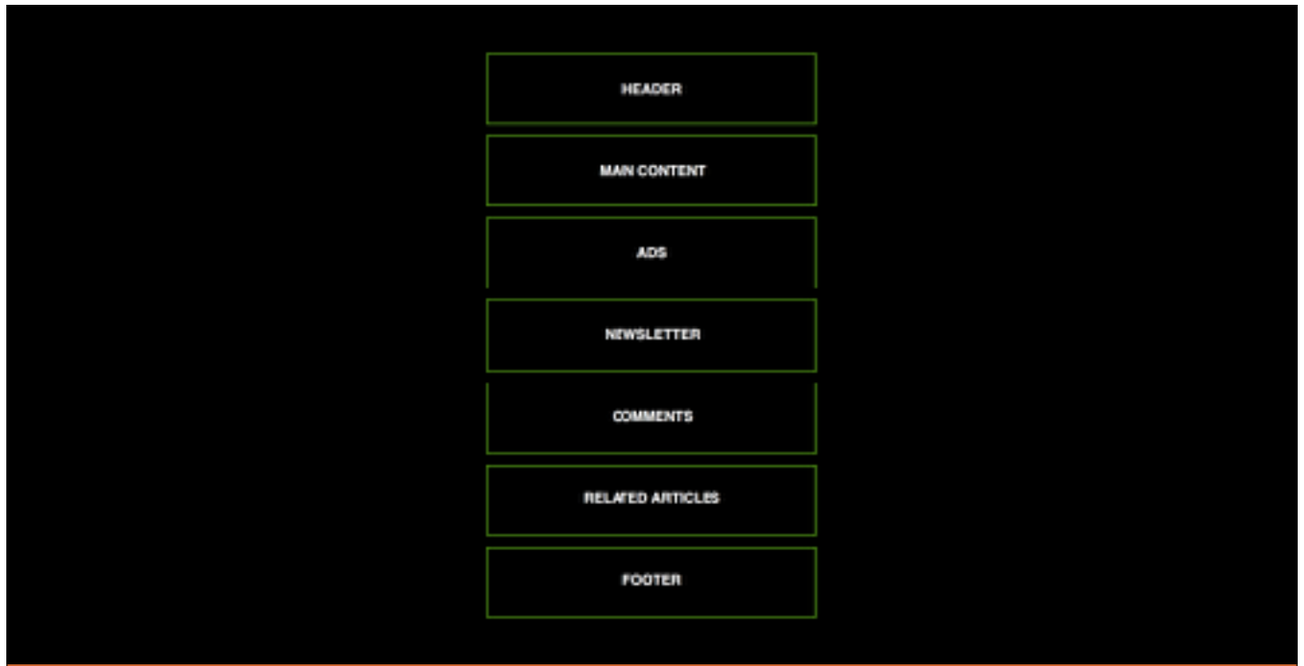
- Auch Screen Reader verwenden mehr Menschen als man glauben würde.
- Allen voran blinde Menschen.
- Aber auch Menschen mit Sehschwächen, die ergänzend einen Screenreader verwenden.
- Menschen mit Lernschwächen.



- Die Visuelle Reihenfolge kann ich auf unterschiedliche Art und Weise beeinflussen, was super ist, aber sich schlecht auf die Accessibility und User Experience im allgemeinen auswirken kann.
- Ich kann leider nicht auf alle im Detail eingehen, habe mir aber eine besondere neue Technik herausgenommen. Wer mehr erfahren möchte, kann und soll mich bitte nachher ausfragen.



Areas ermöglichen es Layouts in CSS abzubilden und zusammenzubauen.



Beispiel: Mobiles Layout.

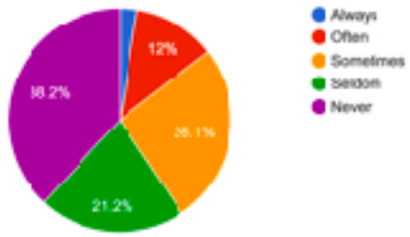
Sagen wir, wir wollen ein einspaltiges Design mit Header, Hauptinhalt, Werbung, Newsletter, Kommentare, verwandten Artikeln und Footer bauen.

Das können sehr schön mit grid-template-areas machen. In Anführungszeichen definiert man jeden einzelnen Bereich in der Seite. Jedes Paar von Anführungszeichen stellt eine neue Reihe dar.

```
$ areas

  body {
    display: grid;
    grid-template-columns: minmax(auto, 480px);
    grid-template-areas:
      "header"
      "newsletter"
      "content"
      "ads"
      "comments"
      "related"
      "footer";
  }
```

Wenn ich oder jemand aus dem marketing auf die Idee kommt, dass der Newsletter gepusht werden und weiter nach oben wandern soll, dann muss ich das HTML nicht angreifen. Ich schiebe einfach die entsprechende Zeile in meiner areas Deklaration nach oben.

- Wenn man die beiden Layouts vergleicht. Links das erste unveränderte und rechts das zweite bei dem die Reihenfolgen nicht mehr zusammenpasst, sieht man dass die Navigation beim zweiten Layout sehr mühsam ist, weil der Fokus hin und her springt.
- Man könnte sich jetzt denken, nagut ok, dass is Mobile, da ist Keyboard-Navigation eh wurscht.

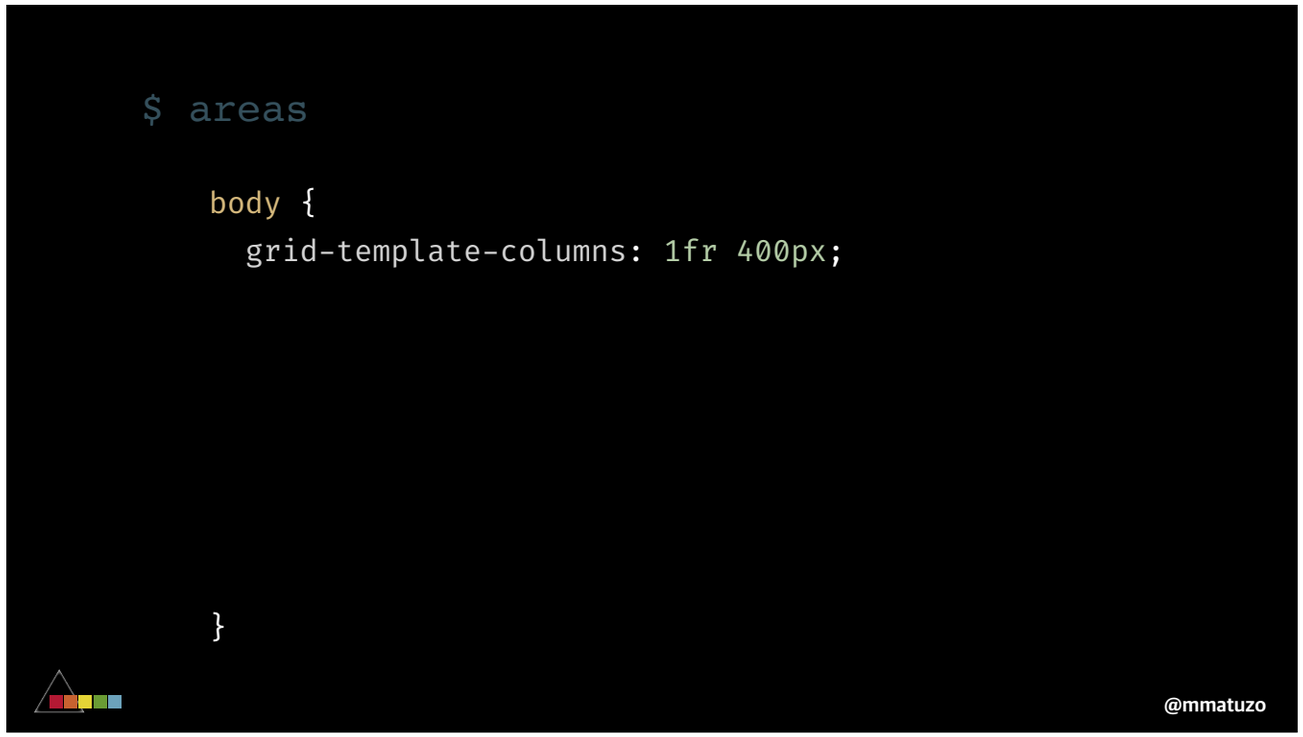


When using a mobile screen reader how often do you use an external keyboard?

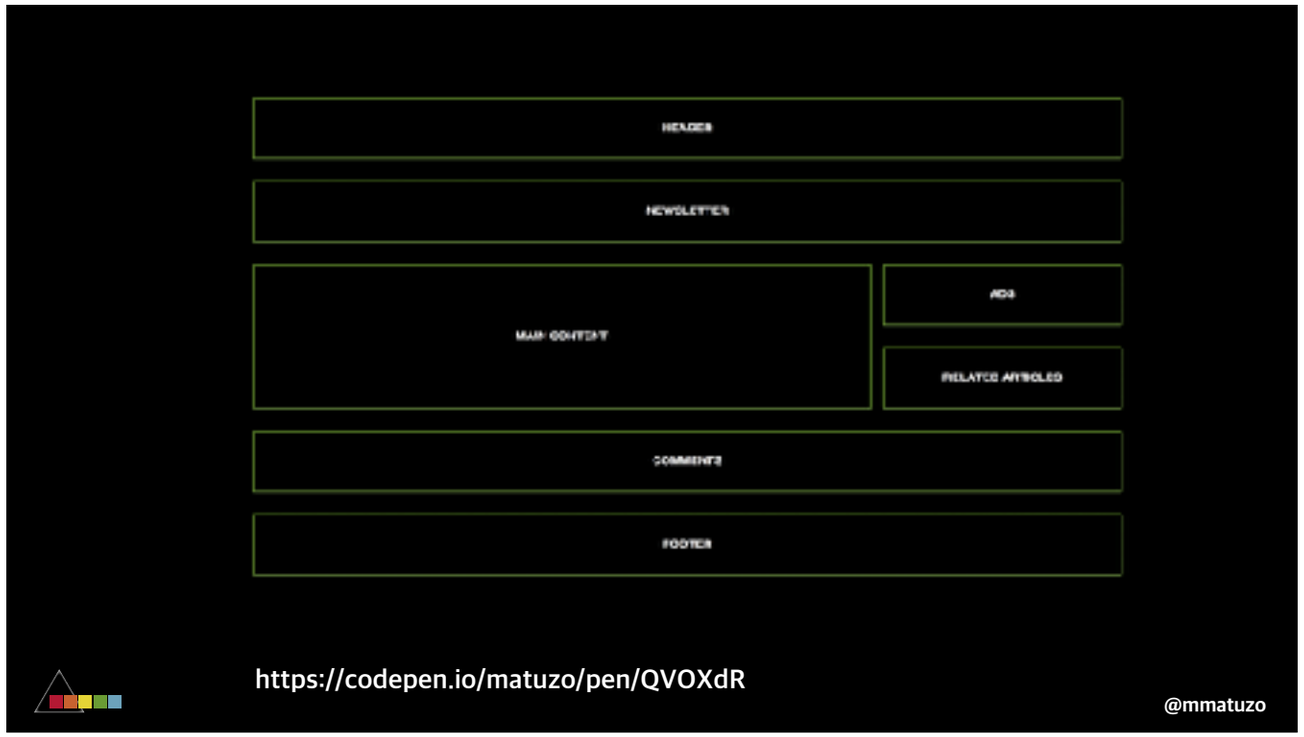
Response	# of Respondents	% of Respondents
Always	59	3.9%
Often	181	11.8%
Sometimes	394	25.7%
Seldom	320	20.9%
Never	577	37.7%

15,7% der befragten Screenreaderuser verwenden immer oder oft ein Keyboard auf ihrem mobilen Endgerät.

Laut dem Screen Reader Survey von 2017 verwenden 15,7% der befragten Screenreaderuser immer oder oft ein Keyboard auf ihrem mobilen Endgerät. 25,7% manchmal.



Außerdem kann ich mit Media Queries Breakpoints setzen und das Layout für breitere Bildschirmgrößen verändern. Mit grid-template-columns mache ich das Layout 2-spaltig. Zwischen den Anführungszeichen ergänze ich einen weiteren Wert. Wenn da „header header“ steht, heißt das, dass der header über zwei Spalten geht. Wenn da „content partners“ steht, heißt das, dass content in dieser Reihe eine Spalte einnimmt und partners die zweite.



Man sieht hier sehr gut, dass die visuelle Reihenfolge überhaupt nicht mehr mit der Reihenfolge im DOM zusammenpasst.

- Jetzt stellt sich nun die Frage, bei wem die Verantwortung liegt. Müssen wir uns darum kümmern oder vielleicht Browserhersteller oder das W3C?
- Ich werde diese Frage nicht beantworten, aber wir können uns mal anschauen, welche Optionen wir haben.

Unsere Optionen

- Source Order dynamisch bei jedem Breakpoint mit JS anpassen. 🙄
- **`tabindex`** oder **`aria-flow-to`** allen Elementen zuweisen und entsprechend ordnen. 😭
- Browsersniffing – Je nach Client/Gerätetyp unterschiedliches HTML ausliefern. 🤢

@mmatuzo

- Was können wir tun?
- Source Order dynamisch bei jedem Breakpoint mit JS anpassen. 🙄
- tabindex oder aria-flow-to allen Elementen zuweisen und entsprechend ordnen. 😭 Sicher nicht.
- Browsersniffing - Je nach Client/Gerätetyp unterschiedliches HTML ausliefern. Wir wissen schon eine Weile, dass das keine gute Idee ist.

Unsere Optionen

- Das Problem mindern indem man Skiplinks zur Verfügung stellt. 😐
- Die visuelle Reihenfolge nicht ändern. 🙄

- Das Problem mindern indem man Skiplinks zur Verfügung stellt. 😐 Jo eh, das sollte wir sowieso machen.
- Die visuelle Reihenfolge nicht ändern. 🙄

„Authors must use order and the grid-placement properties only for visual, not logical, reordering of content.“

- https://drafts.csswg.org/css-grid/#order-accessibility

- Das ist tatsächlich auch was vom W3C empfohlen wird. Wir sollen Reihenfolge nur ändern, wenn es rein um visuelles und nicht logischen Umordnen geht.

Brilliant on the one hand for them trying to at least advise everybody in the right direction but really — come on. […] Suggesting that we don’t all use it just because of this, i think, is wishful thinking.

-Léonie Watson FF Conf 2016 https://www.youtube.com/watch?v=spxT2CmHoPk

Aber es ist einfach unrealistisch, dass Developer das nicht machen, wenn es schon so einfach geht. Insbesondere, wenn es Grid noch viel einfacher macht. Ich habe heute nur eine von mindestens 5 Techniken gezeigt mit denen man Order beeinflussen kann.
Außerdem ist es oft einfach unmöglich die Reihenfolge sinnvoll beizubehalten, vor allem, wenn man sich von einem Breakpoint zum nächsten bewegt.

Was machen wir also?

- Über die Reihenfolge Gedanken machen noch bevor HTML oder CSS geschrieben wird. So früh wie möglich eng mit den DesignerInnen zusammenarbeiten.
- Mit sauber strukturiertem HTML beginnen.
- CSS mobile first schreiben und entsprechend das Layout anpassen.

@mmatuzo

- Also, was machen wir nun?
- Über die Reihenfolge Gedanken machen noch bevor HTML oder CSS geschrieben wird. So früh wie möglich eng mit den DesignerInnen zusammenarbeiten.
- Mit sauber strukturiertem HTML beginnen.
- CSS mobile first schreiben und entsprechend das Layout anpassen.

Was machen wir also?

- Das Layout testen indem man mit der Tab-Taste auf unterschiedlichen Bildschirmgrößen durch die Seiten navigiert.
- Wenn es Diskrepanzen in der Reihenfolge gibt, zurück zum Source Code und das HTML anpassen.

- Das Layout testen indem man mit der Tab-Taste auf unterschiedlichen Bildschirmgrößen durch die Seiten navigiert.
- Wenn es Diskrepanzen in der Reihenfolge gibt, zurück zum Source Code und das HTML anpassen.
- Bevor ich zum nächsten Thema kommen, möchte ich nochmal kurz über Pink Floyd sprechen.

The Dark Side of The Moon war 736 Wochen in den amerikanischen Billboard Charts. Das sind mehr als 14 Jahre!

DOKUMENTSTRUKTUR ABFLACHEN

@mmatuzo

„I believe there will be a strong temptation, especially with Grid, to flatten out document structure in order that all elements become a child of the element with the Grid declared. Making layout simple, but at what cost?“

-Rachel Andrew
(https://rachelandrew.co.uk/archives/2015/07/28/modern-css-layout-power-and-responsibility/)

- Einige Experten haben schon recht früh befürchtet, dass wir verleitet werden, die Dokumentstruktur unserer Seiten abzuflachen, nur um mit Grid arbeiten zu können.
- Warum das so ist, möchte ich mit einem Beispiel illustrieren.

$ dokumentstruktur abflachen

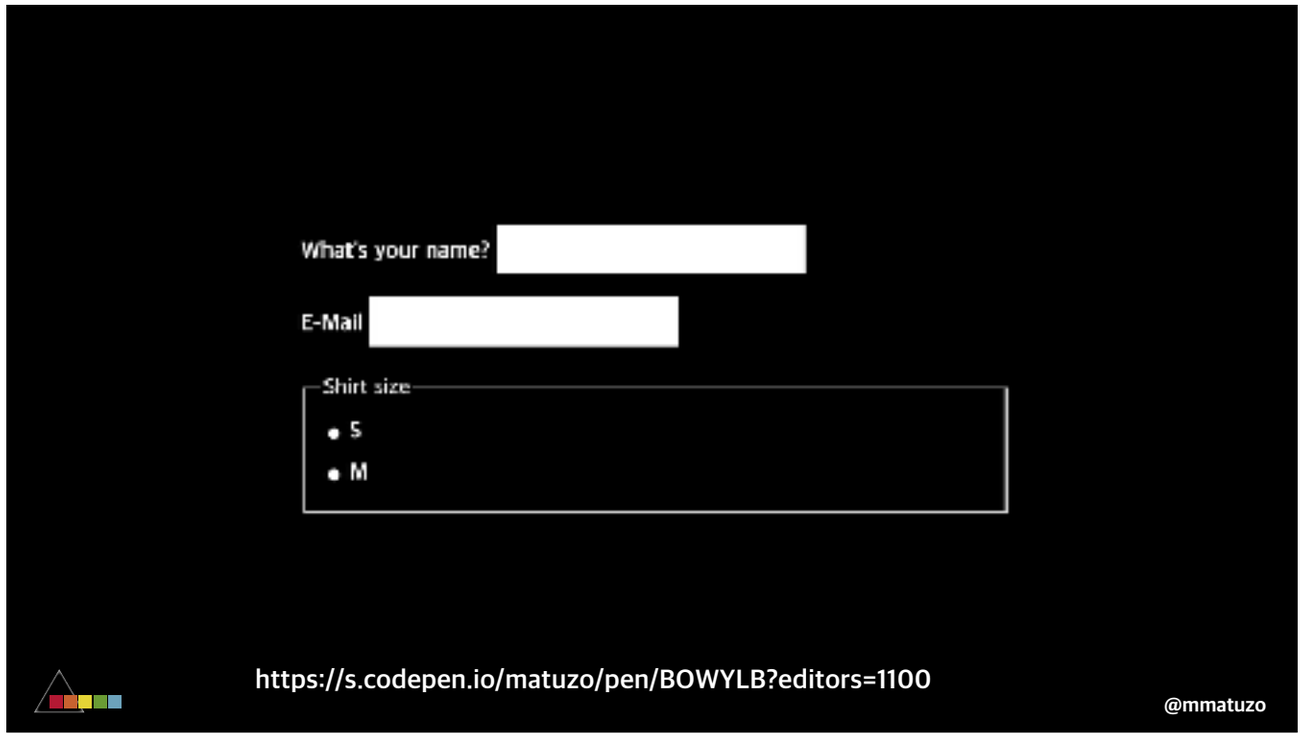
```
<form>
  <div>
    <label for="name">What's your name?</label>
    <input type="text" id="name" />
  </div>
  <div>
    <label for="email">E-Mail</label>
    <input type="email" id="email" />
  </div>
  ...
```

- Sagen wir, wir haben ein Formular mit zwei divs mit jeweils einem Label und einem text-Input. Einmal für den Vornamen und einmal für die E-Mail Adresse.

$ dokumentstruktur abflachen

```
      …
      <fieldset>
        <legend>Shirt size</legend>
        <div>
          <input type="radio" id="s" name="r">
          <label for="s">S</label>
        </div>
        <div>
          <input type="radio" id="m" name="r">
          <label for="m">M</label>
        </div>
      </fieldset>
    </form>
```

Und dann noch ein fieldset mit einem legend Element für die T-Shirtgröße und zwei Radiobuttons mit den Labels S und M.



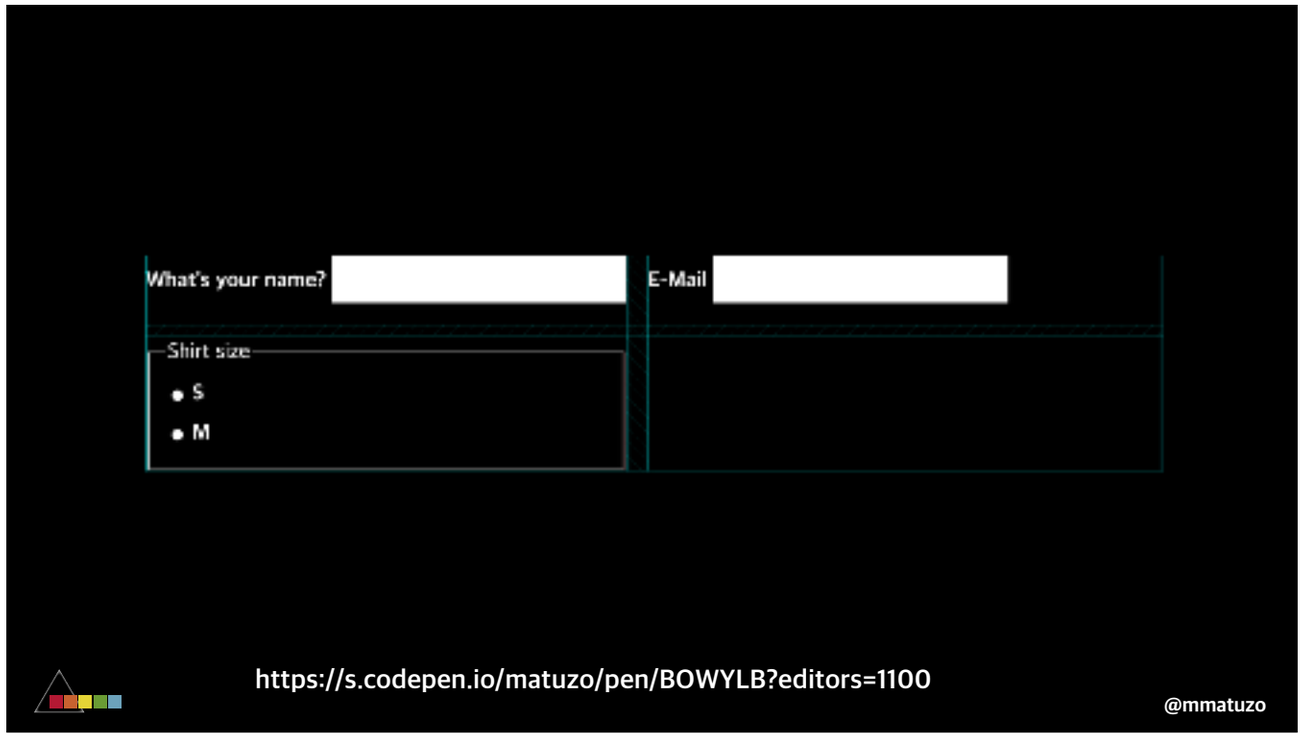
Das sieht so aus. Es wäre aber nett, wenn alle drei Labels bündig abschließen würden.

$ dokumentstruktur abflachen

```
form {
  display: grid;
  grid-template-columns: max-content minmax(auto, 600px);
  grid-gap: 10px 20px;
}
```

@mmatuzo

Kein Ding mit Grid. Ich mache aus dem Formular ein Grid-Container, erstelle zwei Spalten. Mit max-content schaffe ich es, dass die Spalte so breit ist, wie das breiteste Label. Mit grid-gap füge ich einen Abstand zwischen den Spalten hinzu.



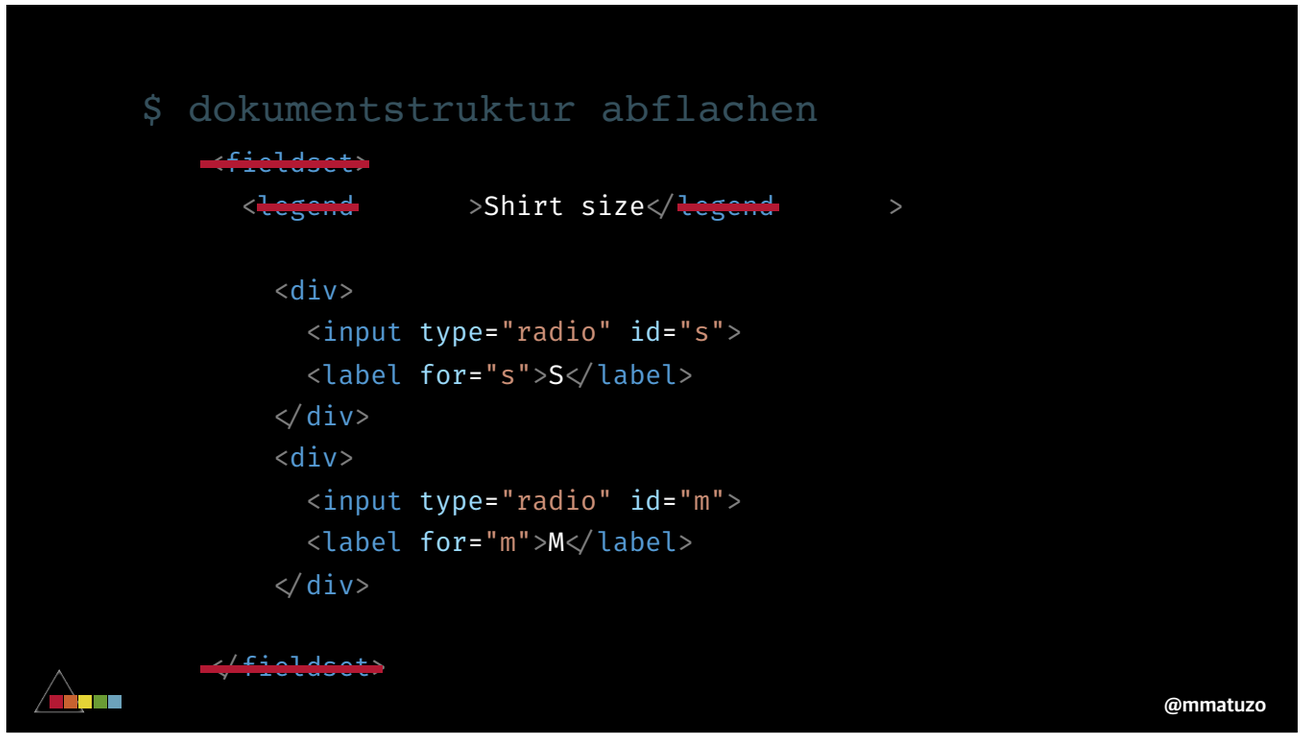
Sieht nicht wie erwartet aus. Das erste div ist in der ersten Spalten, das zwei in der zweiten Spalte und das fieldset in der ersten Spalte in der zweiten Reihe.
Das liegt daran, dass nur direkte Kindelemente in einem Grid platziert werden, also die beiden divs und das fieldset, aber nicht die Label, legend und Input Elemente.

$ dokumentstruktur abflachen

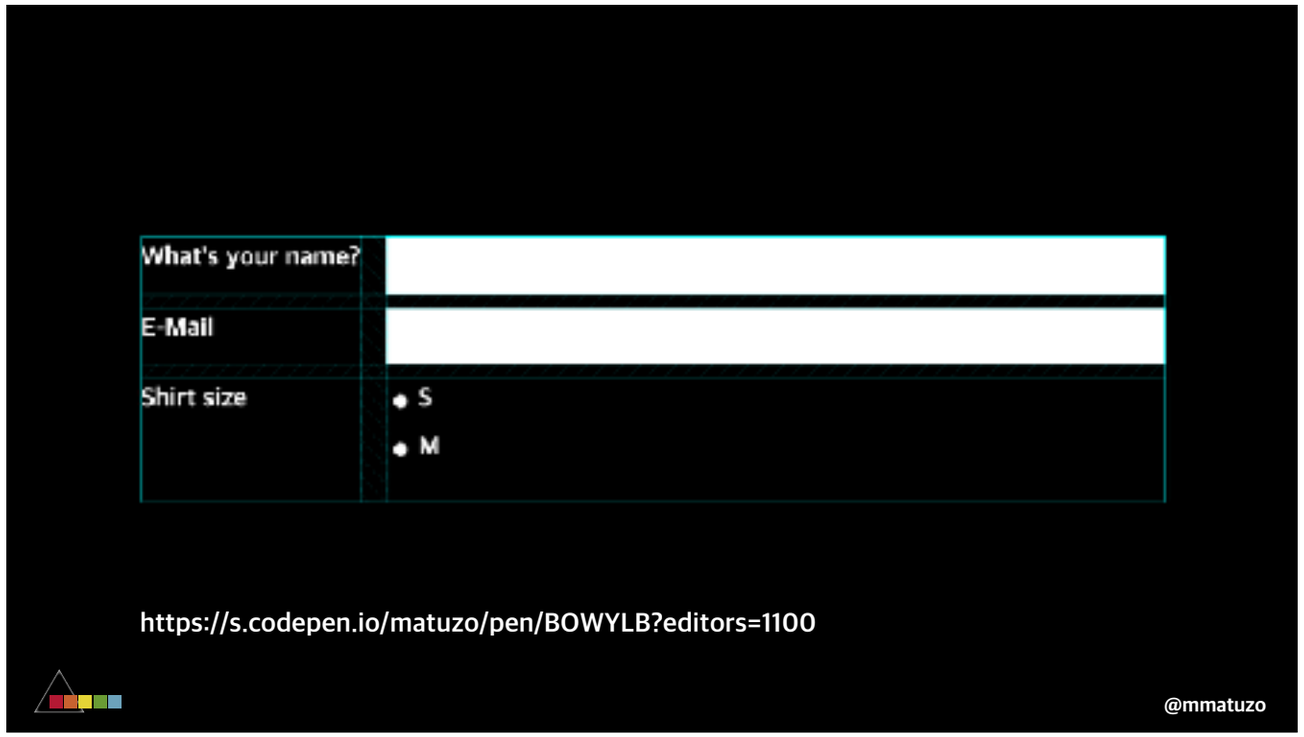
```
<div>
  <label for="name">What's your name?</label>
  <input type="text" id="name" />
</div>
<div>
  <label for="email">E-Mail</label>
  <input type="email" id="email" />
</div>
```

@mmatuzo

Das lässt sich einfach lösen. Einfach die divs entfernen.



Das fieldset kommt auch weg und aus dem legend mache ich ein strong.



Und das Ergebnis ist optisch perfekt, aber wir haben so nicht nur auf semantische Elemente verzichtet, unser Dokument ist bei der Darstellung ohne CSS auch viel schlechter zu konsumieren.

Darstellung des Formulars ohne CSS

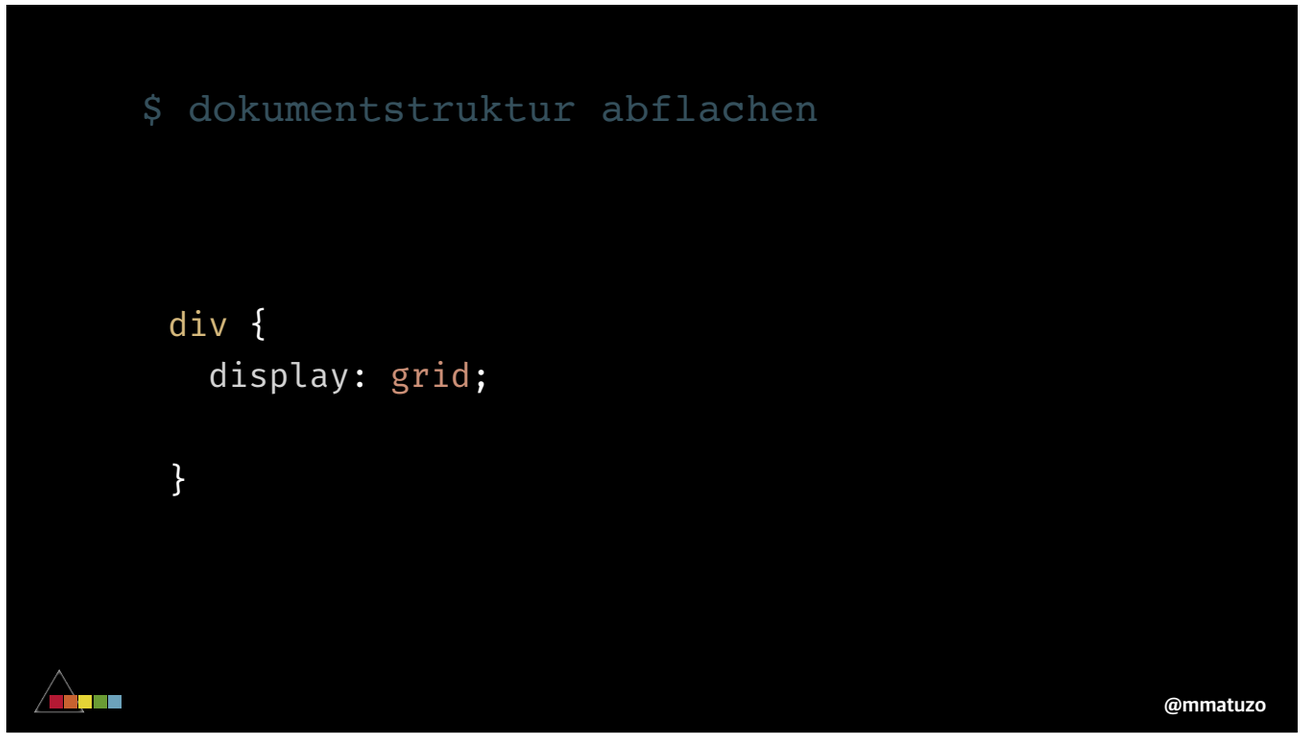
- Und das passiert auch im echten Leben. Ich war schon oft genug auf Urlaub mit sehr schlechter Internetverbindung und habe nur HTML-Versionen einer Website bekommen, weil es das CSS nicht über die Leitung geschafft hat.
Was ist die Lösung?

Subgrid!

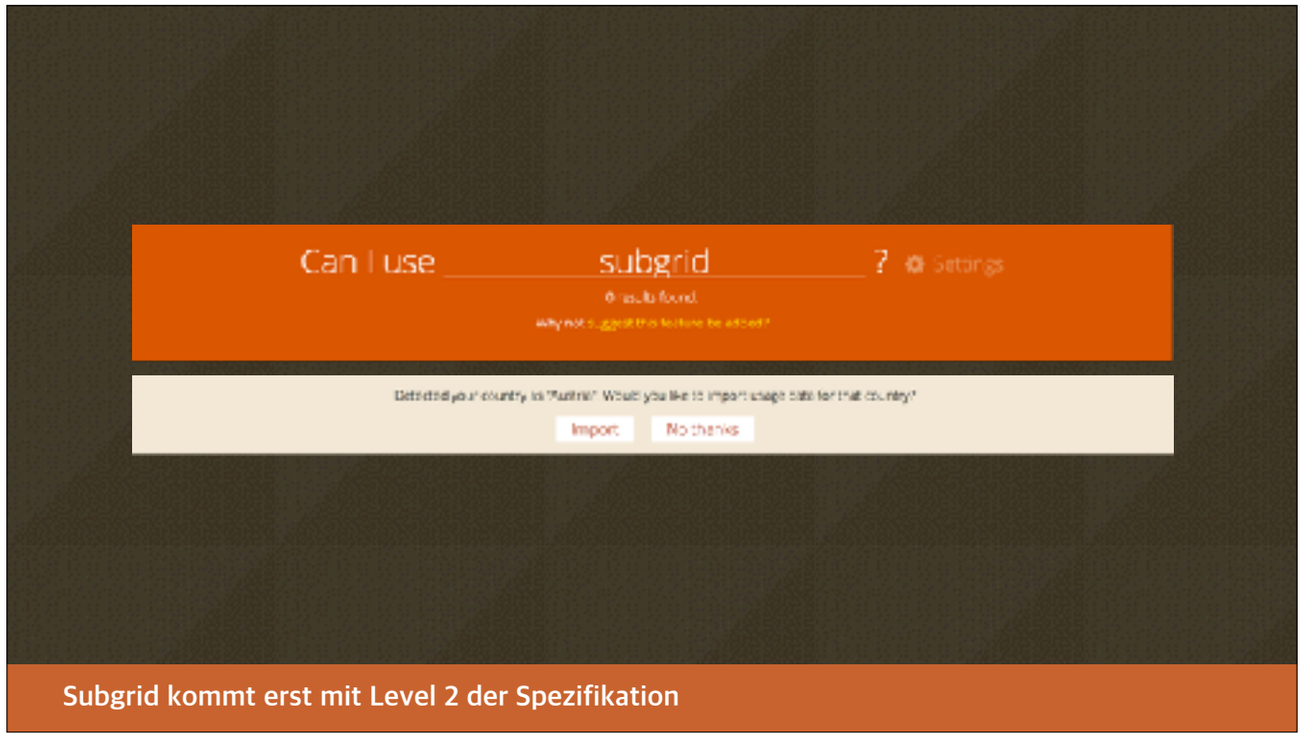
„A grid container that is itself a grid item can defer the definition of its rows and columns to its parent grid container, making it a subgrid.“

- https://drafts.csswg.org/css-grid-2/

Mit Subgrid ist es möglich, dass ein Grid-Kindelement seine Grid-Eigenschaften an die eigenen direkten Kindelemente überträgt, sodass diese Teil des Grid werden.



Das kann dann so aussehen. Anstatt, dass man wieder Spalten und Reihen definiert, gibt man den Wert „subgrid“ in der Spalten oder Reihendeklaration an.



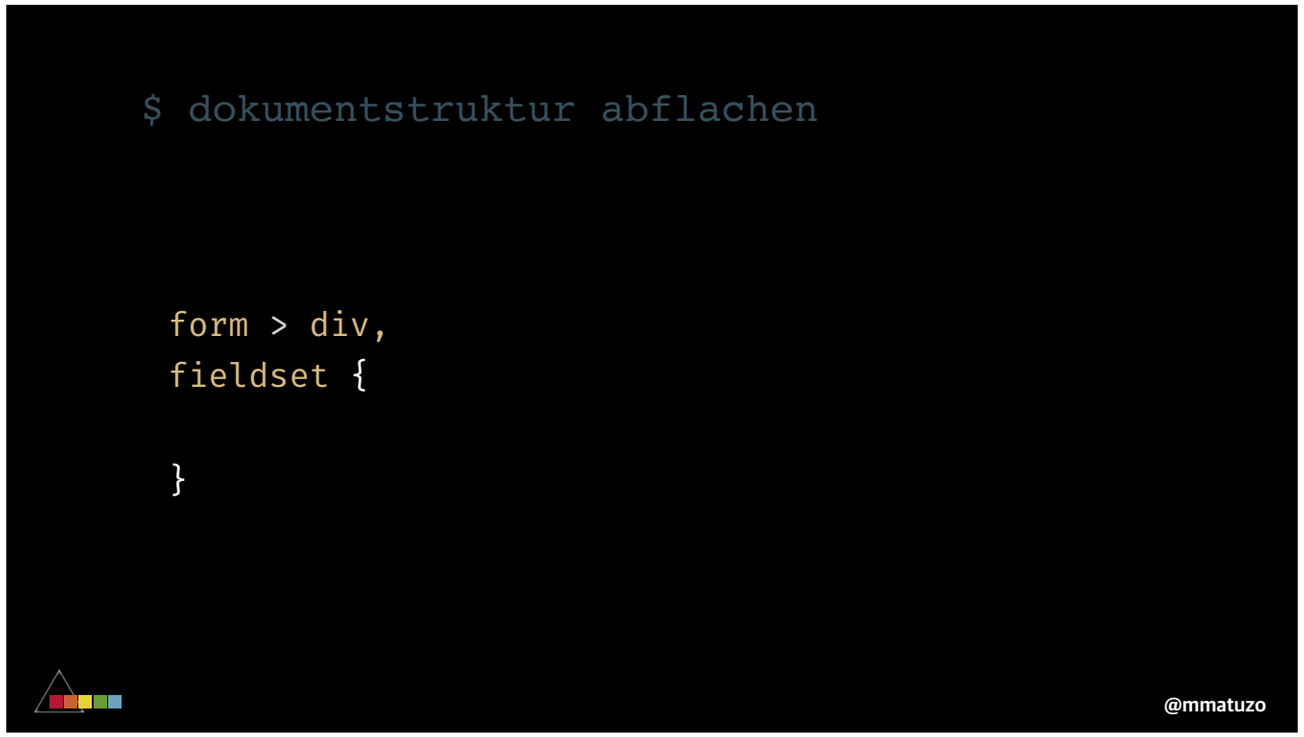
Subgrid kommt erst mit Level 2 der Spezifikation

Der Support für Subgrid ist nur leider so schlecht, dass es nicht mal eine Seite auf caniuse.com gibt. Das liegt daran, dass Subgrid erst mit Level 2 der Spezifikation kommt.

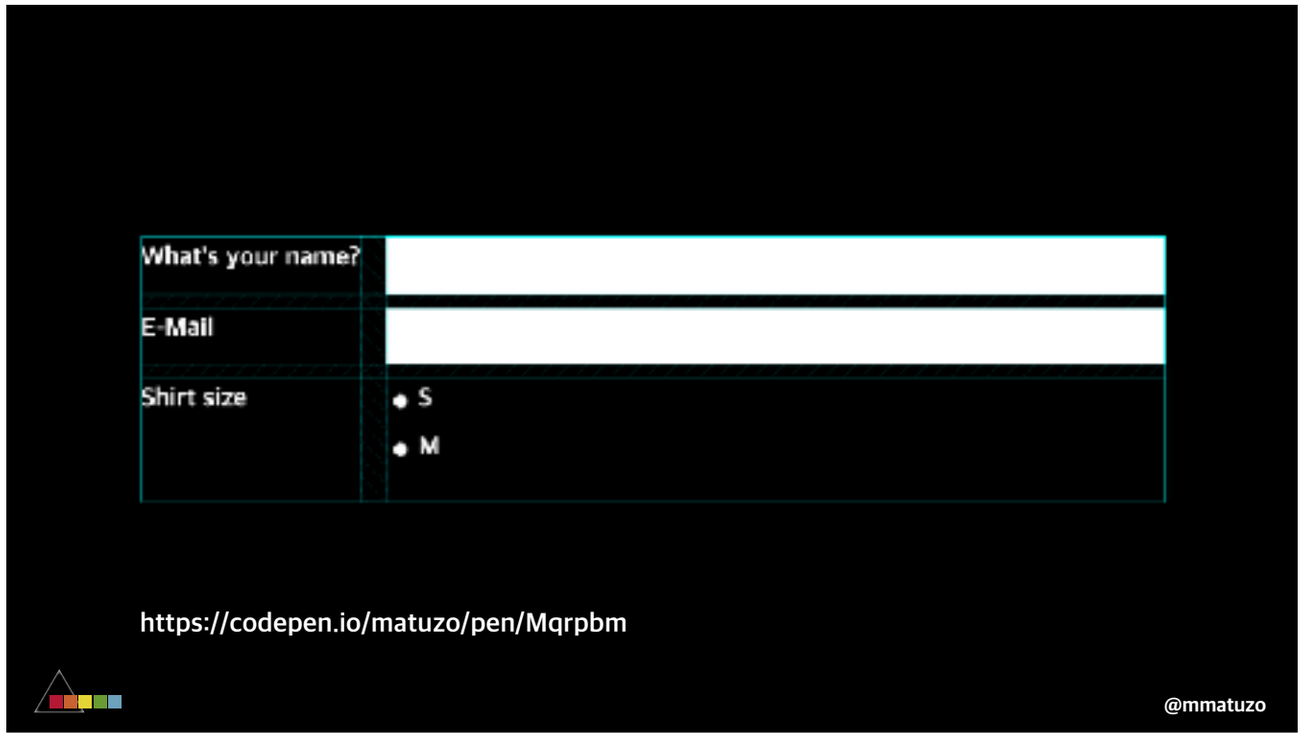
„`display: contents` causes an element's children to appear as if they were direct children of the element's parent, ignoring the element itself“

- https://caniuse.com/#search=contents

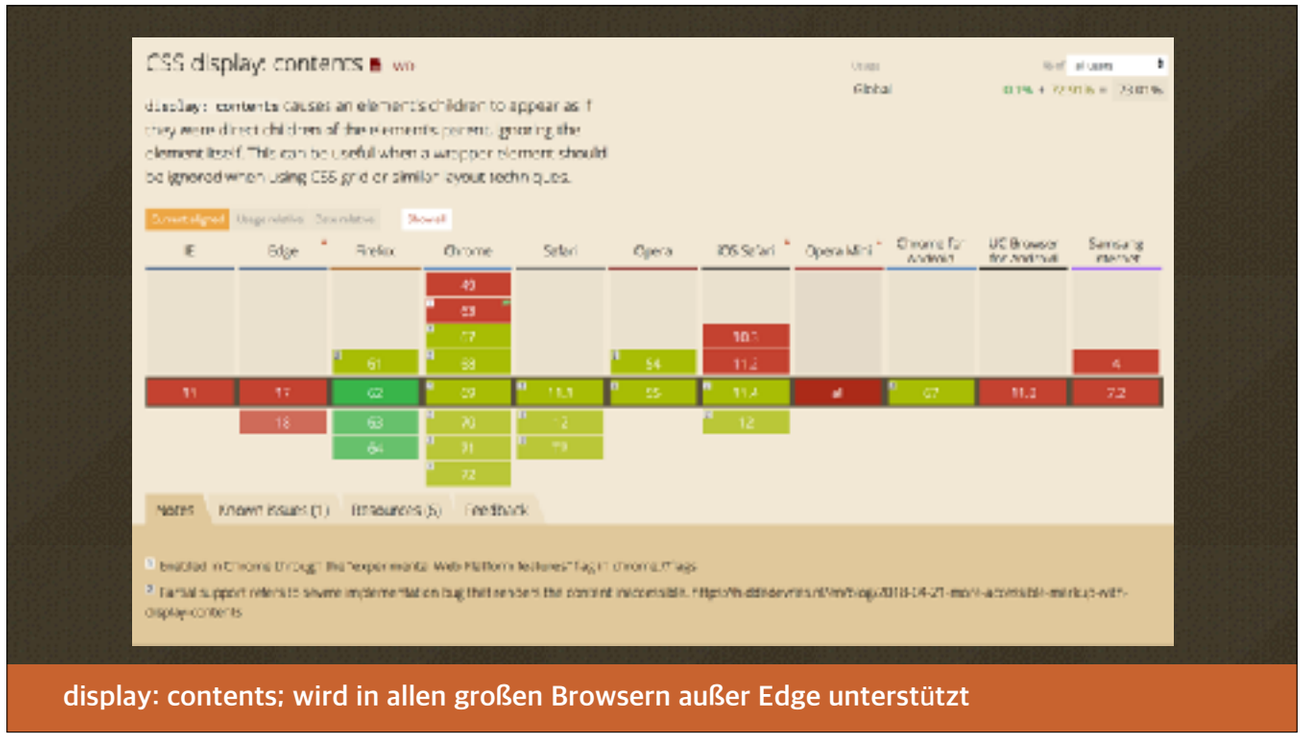
In der Zwischenzeit könnten wir display: contents verwenden. Wenn ein Elemente display: contents hat, tun seine Kindelemente so als würd das Elemente nicht existieren.



Geben wir unseren divs und dem fieldset display: contents, sieht unser Formular so aus.



Perfekt! Genau, was wir wollten.



display: contents; wird in allen großen Browsern außer Edge unterstützt

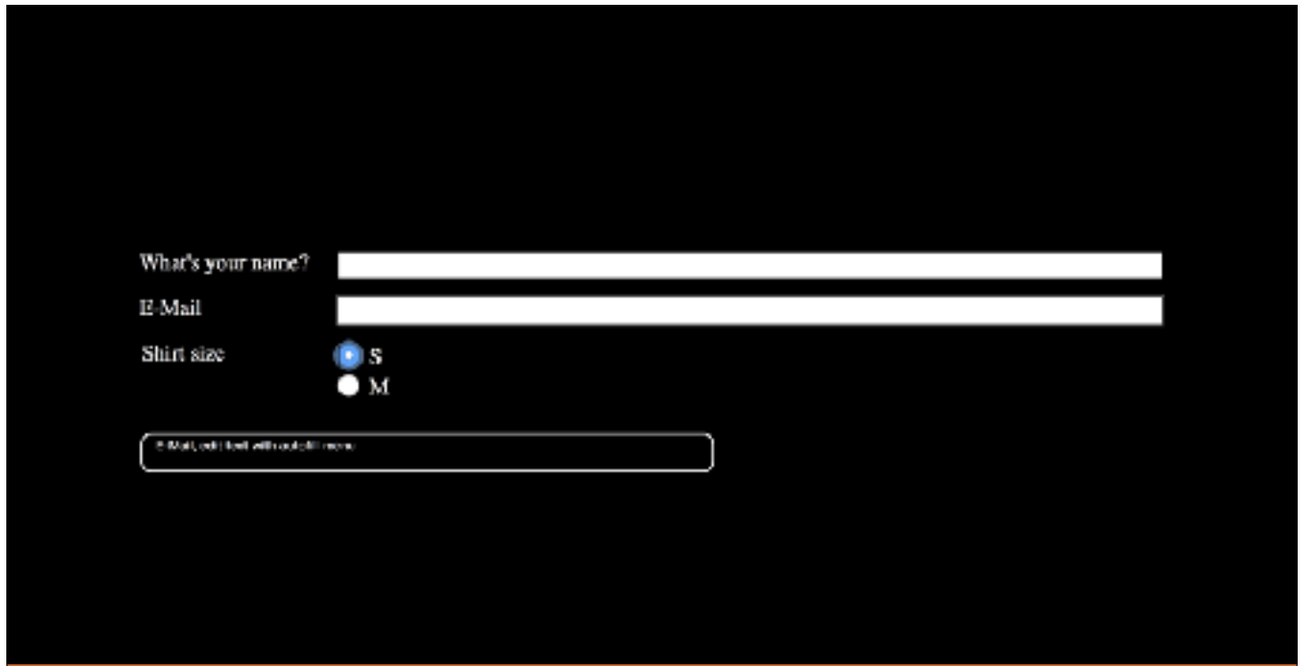
Support ist ziemlich gut, einzig Internet Explorer und Edge stellen noch ein Problem da.

Wegen eines Bugs entfernt display: contents; Elemente aus dem Accessibility-Tree.

Ein viel größeres Problem aktuell ist viel mehr aber, dass es einen Bug gibt. Und zwar ist es in den meisten Browser aktuell so, dass display: contents das Element auf dem es angewendet wird auch aus dem Accessibility Tree entfernt.
In unserem konkreten Beispiel macht das bei den divs keinen Unterschied, aber beim fieldset.

Ohne display: contents; wird das legend Element vorgelesen.

Hier verwende ich VoiceOver um das Fieldset vorzulesen, ohne dass ich display contents angewendet habe. Man hört, dass es mit dem Label auch der Wert des legend Elements vorgelesen wird.



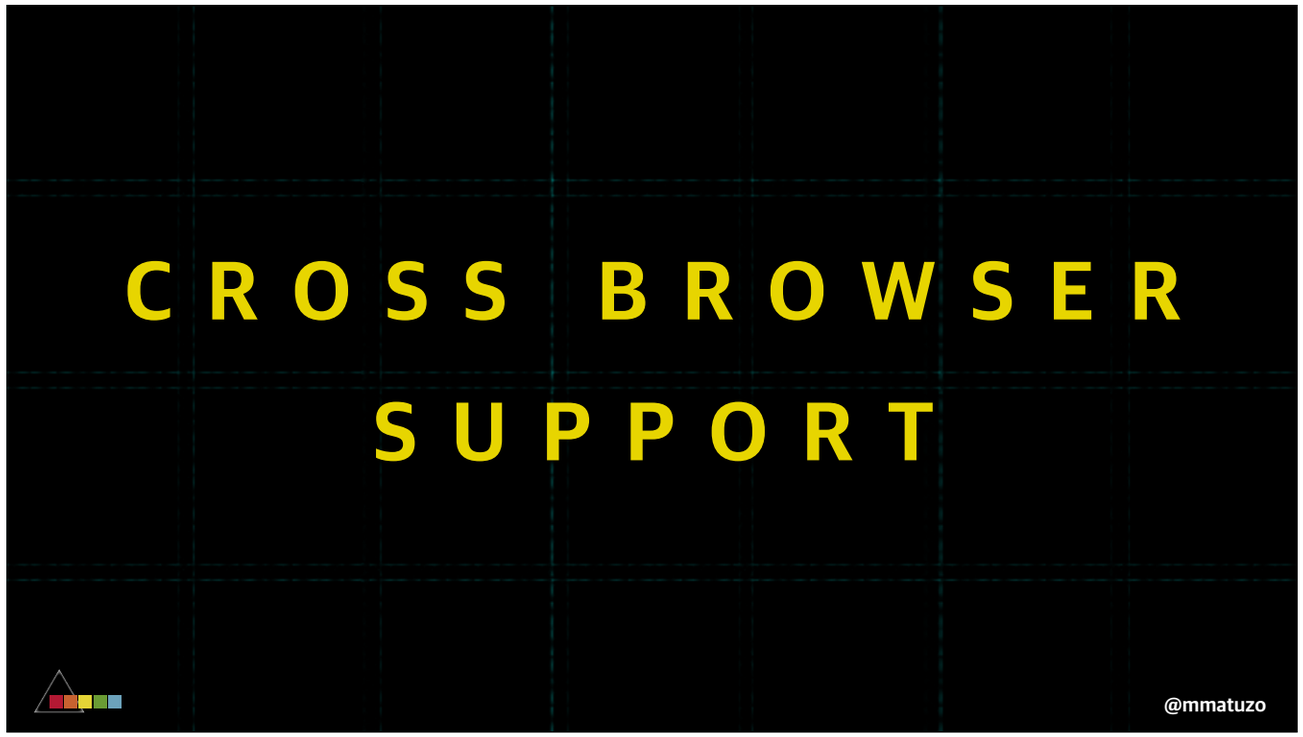
Mit display: contents; wird das legend Element nicht vorgelesen.

Jetzt das gleiche nur mit display: contents. Man hört nur das Label, weil das fieldset und damit auch das legend Element jetzt nicht mehr im Accessibility Tree sind.

Diese Eigenschaft kann nicht nur im Grid Kontext praktisch und problematisch sein. display: contents wird wohl schon als Reset verwendet. Zum Beispiel bei einer Liste um padding, margin und den list-type zu entfernen. Super praktisch, aber aktuell leider sehr schlecht für die Accessibility.

- Pink Floyd haben 1975 dabei geholfen, den Monty Python Film
„Die Ritter der Kokosnuß“ zu finanzieren.

Andere Investoren waren unter anderem Led Zeppelin und Genesis.
- Soweit ich weiß ging es da um steuerliche Vorteile.



- Das nächste Thema ist Browser Support

Kurze Wiederholung der Geschichte

- 2011: Microsoft liefert die erste Implementierung von Grid hinter dem `-ms-` Browserprefix in IE 10.
- Chrome, Firefox, Opera, Safari und Edge implementieren Grid im Laufe des Jahres 2017.

- Ich habe bereits erwähnt, dass Microsoft eine Variante der Grid Spezifikation 2011 veröffentlicht haben.
- Alle anderen Browser eine Komplet andere Version 6 Jahre später.

Eigenschaften: Level 1 vs. MS Spezifikation

CR Level 1 Eigenschaft	IE10 Implementation
grid-template-columns	
grid-template-rows	
grid-template-areas	
grid-template	
grid-auto-columns	
grid-auto-rows	
grid-auto-flow	
grid	
grid-row-start	
grid-column-start	
grid-row-end	
grid-column-end	

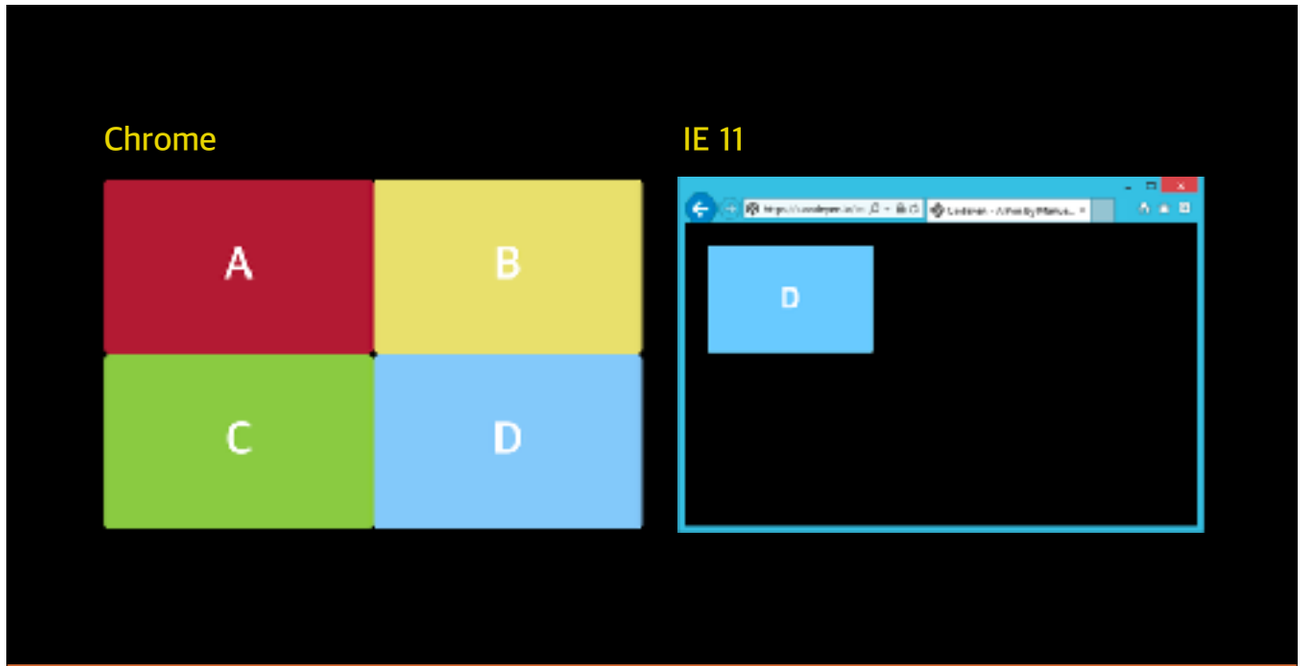
CR Level 1 Eigenschaft	IE10 Implementation
grid-row	
grid-column	
grid-area	
grid-row-gap	
grid-column-gap	
grid-auto-flow	
grid-gap	
-	
-	
align-self	
justify-self	

@mmatuzo

- Wenn man die beiden Spezifikationen vergleicht, sieht man dass die ursprüngliche Version einen Großteil der Dinge, die heute möglich sind, nicht kann.
- Das Erstellen von Spalten und Reihen ist möglich, das Platzieren auch, aber das wars dann schon. Es gibt kein Auto-Placement, keine Areas und keine Gaps.

```
$ browser support

    .grid {




      display: grid;
      grid-template-columns: 200px 200px;
      grid-template-rows: 130px 130px;
    }
```

- Hier ein kurzes Beispiel
- Ich kann die display Eigenschaft, sowie grid-template-columns und grid-template-rows prefixen.

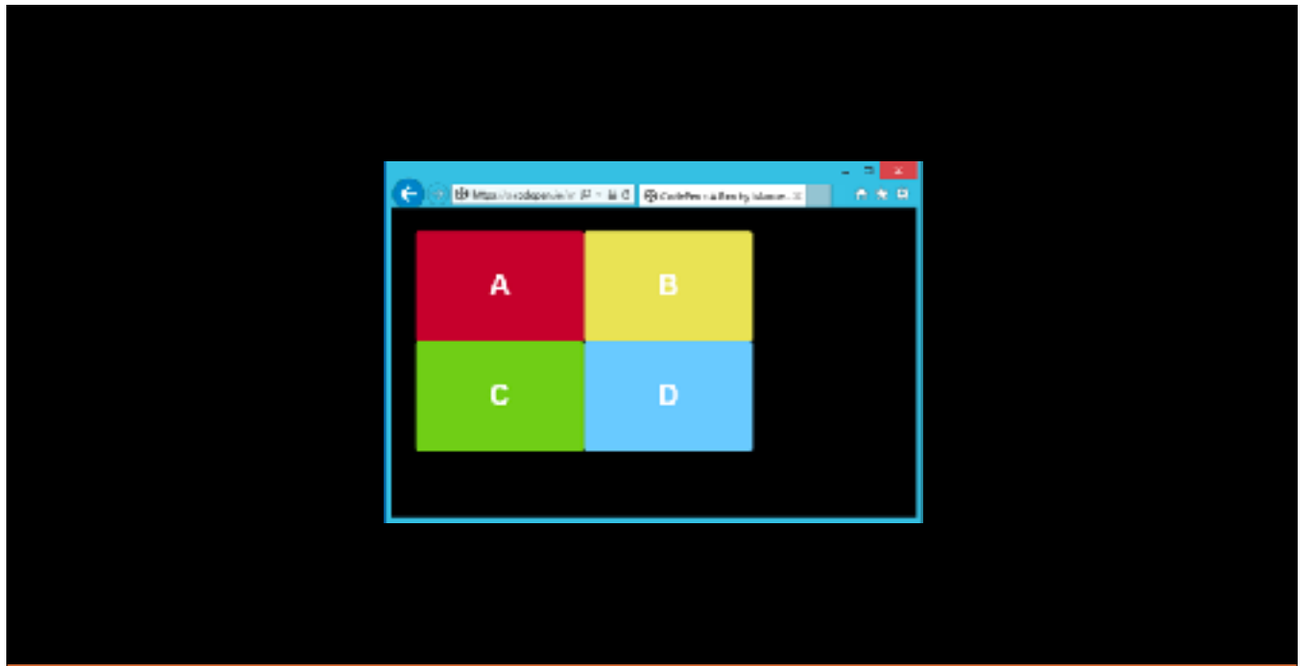


4 Elemente in einem 2 x 2 Grid ohne explizite Platzierung: Darstellung in Chrome und IE11

- In Chrome bekomme ich in diesem konkreten Beispiel ein 2 Mal 2 Grid in dem meine vier Elemente platziert werden.
- In IE 11 sieht das anders aus. Da bekomme ich zwar auch ein Grid aber alle Elemente stapeln sich an einer Stelle.

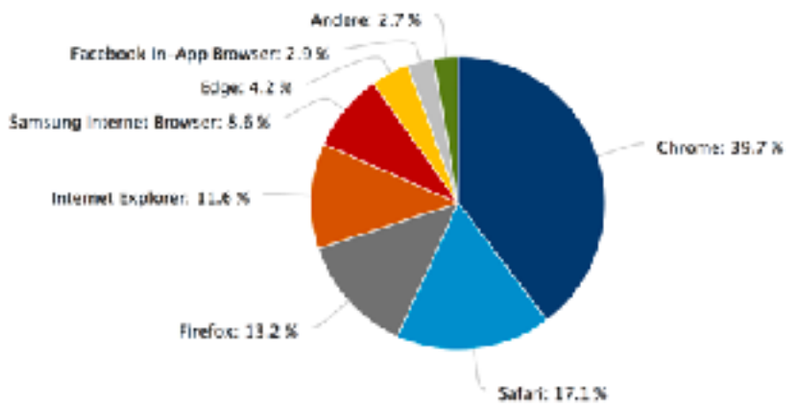
```
$ browser support

  .item:nth-child(1) {
    -ms-grid-column: 1;
    -ms-grid-row: 1;
  }
```

- Grund ist eben, dass es kein Auto-Placement gibt.
- Man muss jedes Element separat angreifen und entsprechend positionieren.



4 Elemente in einem 2 x 2 Grid mit expliziter Platzierung: Darstellung in IE11

Wenn man das macht, sieht auch in IE 11 das Grid so aus wie man es sich vorstellt.



11,6% IE Nutzer in Österreich.

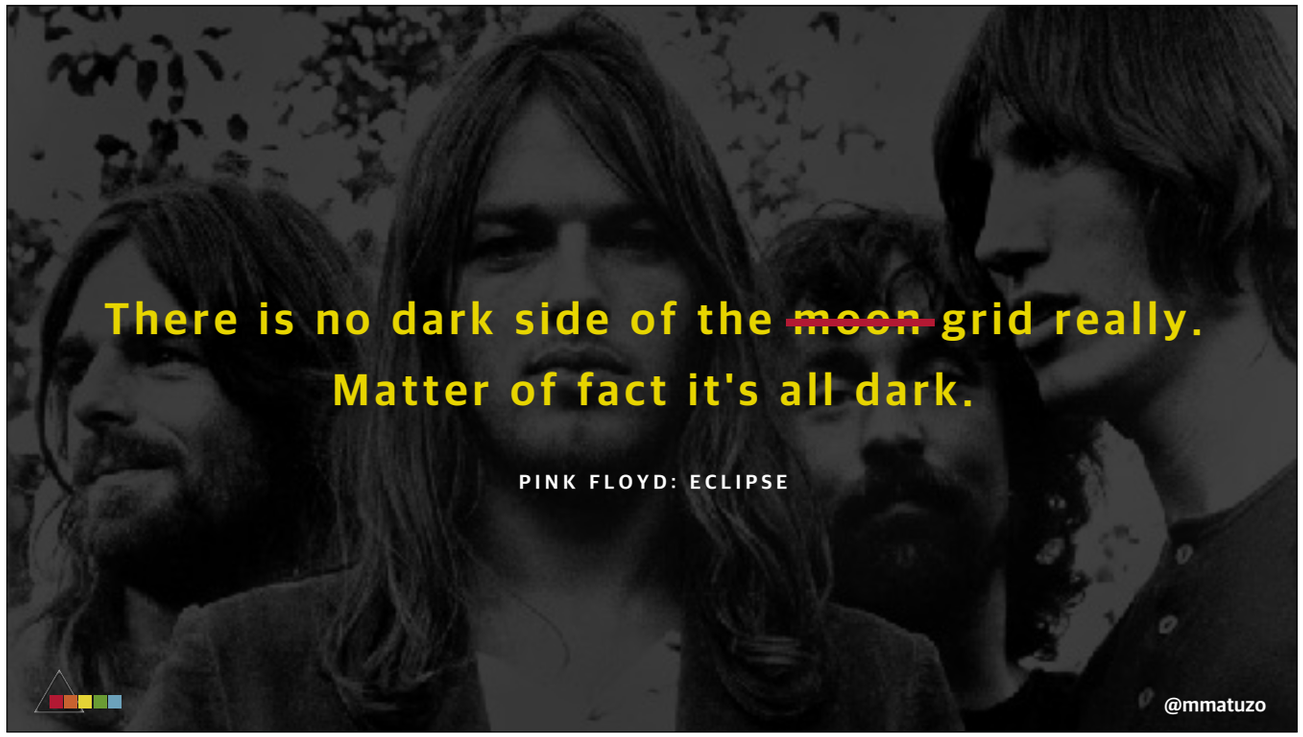
- Warum spreche ich überhaupt über IE10 und IE11? Wir haben 2018.
- Naja, offensichtlich verwenden noch knapp 12 Prozent der User in Österreich Internet Explorer.
- Solche Statistiken sind immer schwierig, ich weiß, aber auch wenns nur ansatzweise 10 Prozent sind, ist das viel.

Internet Explorer Unterstützung stellt für viele EntwicklerInnen noch eine große Hürde da.

- Anscheinend dürfte IE Support für einige Leute Grund dafür sein Grid noch nicht zu verwenden.
- In einer Umfrage auf Twitter haben 30% von 441 TeilnehmerInnen angegeben, dass sie Grid noch nicht wegen dem schlechten IE Support verwenden.

> „I don’t think we owe it to any users to make it all exactly the same. Therefore we can get away with keeping fallbacks very simple. My hypothesis: users don’t mind, they’ve come for the content.“
>
> -Hidde de Vries
> https://hiddedevries.nl/en/blog/2018-08-11-lets-serve-everyone-good-looking-content

Ich glaube, dass das falsch ist. Wir sollten versuchen alle die tollen Techniken, die wir uns schon immer gewünscht haben, auch zu nutzen. Wir leben in einer Zeit in der eine Website nicht überall gleich aussehen muss. Unsere Kunden haben das mittlerweile auch gelernt. Es ist 2018, Responsive Webdesign gibt es seit 2010. Es gibt einfach nicht mehr die „Genau so sieht die Website aus“-Version. Es ist sehr stark Kontext-abhängig. Websites sehen je nach Browser, Betriebssystem, Bildschirmgröße und Gerätetyp anders aus.



- Ich habe nun einige negative Dinge gezeigt, bei weitem nicht alle. Was heißt das jetzt fürs uns? Grid verwenden oder nicht verwenden?
- Ich möchte diese Frage mit einem Zitat beantworten.
- „ There is no dark side of the moon grid really. Matter of fact it's all dark.“
- Alles ist düster. Nein, natürlich nicht. Wir müssen nur verantwortlich mit Grid umgehen.

Verantwortlicher Umgang mit Grid

- Eigenschaften wie `grid-auto-flow: dense;` oder andere Techniken, die die visuelle Reihenfolge verändern, bei interaktionsintensiven Komponenten vermeiden.
- Komponenten mit dem Keyboard testen.
- Nicht die Semantik verschlechtern und die Dokumentstruktur abflachen.

@mmatuzo

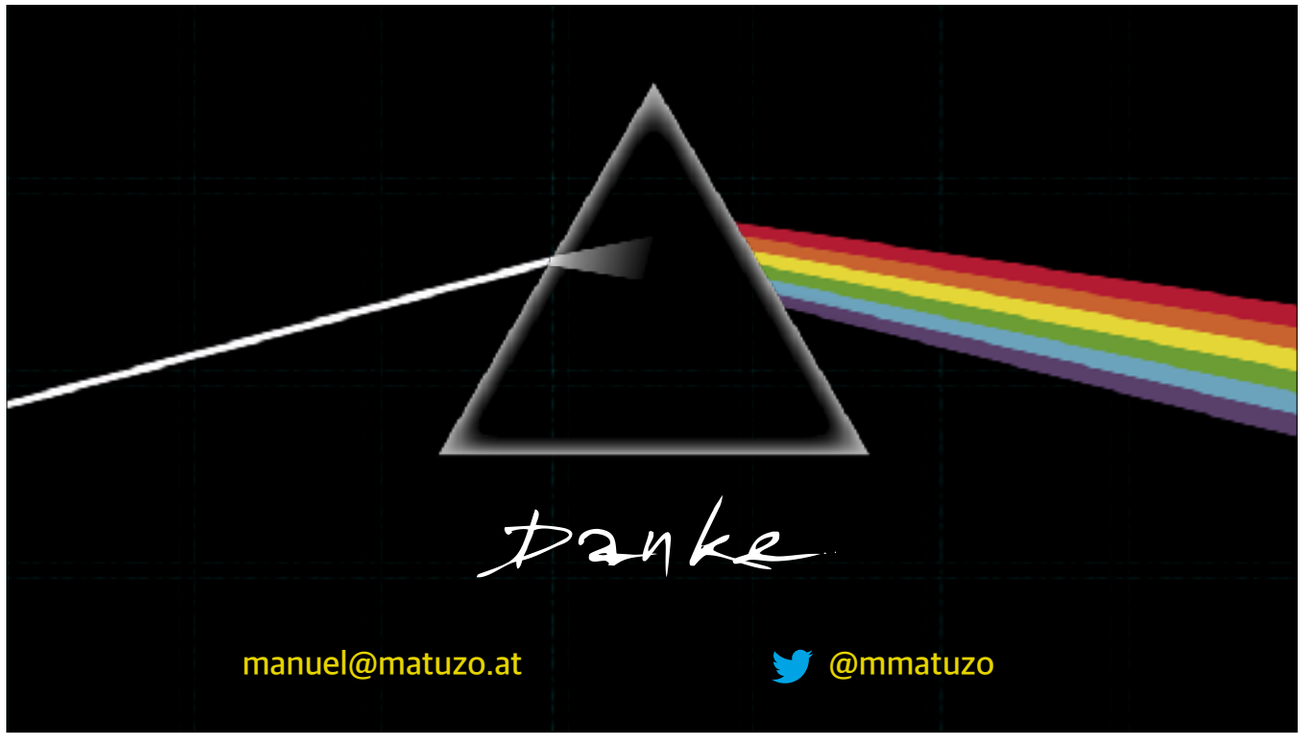
- Eigenschaften wie grid-auto-flow: dense; oder andere Techniken, die die visuelle Reihenfolge verändern, bei interaktionsintensiven Komponenten vermeiden.
- Komponenten mit dem Keyboard testen.
- Nicht die Semantik verschlechtern und die Dokumentstruktur abflachen.

Verantwortlicher Umgang mit Grid

- Wenn man Autoprefixer verwendet, dann so, dass zu jeder Zeit für jede/n im Team klar ist, was warum passiert.
- Sicher gehen, dass die Komponenten auch in älteren / eingeschränkteren Browsern funktionieren.
- Die Macht von Progressive Enhancement ausnutzen.

@mmatuzo

- Wenn man Autoprefixer verwendet, dann so, dass zu jeder Zeit für jede/n im Team klar ist, was warum passiert.
- Sicher gehen, dass die Komponenten auch in älteren / eingeschränkteren Browsern funktionieren.
- Die Macht von Progressive Enhancement ausnutzen.
- Das war natürlich nicht alles. Ich habe noch viel mehr Material. Wenn jemand mehr über Grid wissen möchte, bitte sucht mich in den Pausen oder bei der Aftershow und fragt ruhig alles, was ihr möchtet.



- Vielen Dank für die Aufmerksamkeit.