Indisoft

Responsives Webdesign

04.09.2018

Dozent erste Woche: Elmar Becker (ist für Doku wichtig)

Elementgrößen

Prozent des Einzelelements=Zielgröße/Kontext\*100

Z.B. Navi Seite  
300/1080=27,77777778%  
  
Prinzipiell Wrapper 90%, damit angenehmer Seitenabstand  
  
Man kann für übergroße Bildschirme zunächst einen max-width definieren, damit die Zeilen nicht allzu arg auseinander gezogen werden ….

#wrapper{  
 max-width:1080;  
 width:90%;

}

Background-Komponenten:   
Als Oberdefinition:  
background: transparent url(images/bild.jpg) 750px bottom no-repeat;  
Die Breitenangabe gibt hier die gesamte Leerfläche des Containers an, nicht aber die einzelnen Seitenelemente

background-size:100% auto; x-Achse 100%, y-Achse auto  
background-size:100% 100% in beiden Achsen 100%, verzerrt dann aber;  
background-size:cover; das Bild füllt den Rahmen  
background-size: contain: Das Bild füllt einen der beiden Achsen, die andere wird autoformatiert.

Bilder formatieren:  
Entweder gibt man dem Rahmen eine eigene id/class, oder aber man versieht das Bild selbst mit eigener id. Dann

max-width: 100%;  
height: auto;

Damit bei Umbrüchen die padding-Abstände nicht stets zur width dazugerechnet werden, gibt man an:  
box-sizing:border-box;  
Hiermit werden die padding-Abstände hinzugerechnet …

CSS-Klasse first-of-type nimmt sich des ersten Kindelements des gleichen Namens an, das ein Tag hat

Vidoetraining:  
hochauflösende Fotos …  
Ethan Marcot:  
RD von responsiver Architecture  
flexible Raster, Medien, Queries  
Die Information steht im Mittelpunkt, keine Informationen völlig verstecken!  
Kompakte Navigationen und flache Hierarchien in der Navigation

Bei alternativen Versionen (Desktop/Smartphone) muss ein Link in der Startseite vorhanden sein, wo es hingehen soll. Weiterhin muss von Desktop zu Smartphone und umgekehrt verlinkt sein.  
Responsive Webdesign umgeht die Schwierigkeiten, daher präferiert Google RD.  
Es gibt noch adaptives Design, hier wird je nach Gerät das Design ausgewählt. Meist werden nur Komponenten adaptiv ausgetauscht, z.B. Medienfiles. (Hybrid, RESS = Responsive + Server Side),   
adaptives Layout ist recht statisch, weil immer von einer zur anderen Größendimension gesprungen wird. Letztlich ist es auch Platzverschwendung, da immer Flächen, die nicht bei den Größen passen, leer bleiben.  
Vorteil: es ist unkomplizierter.

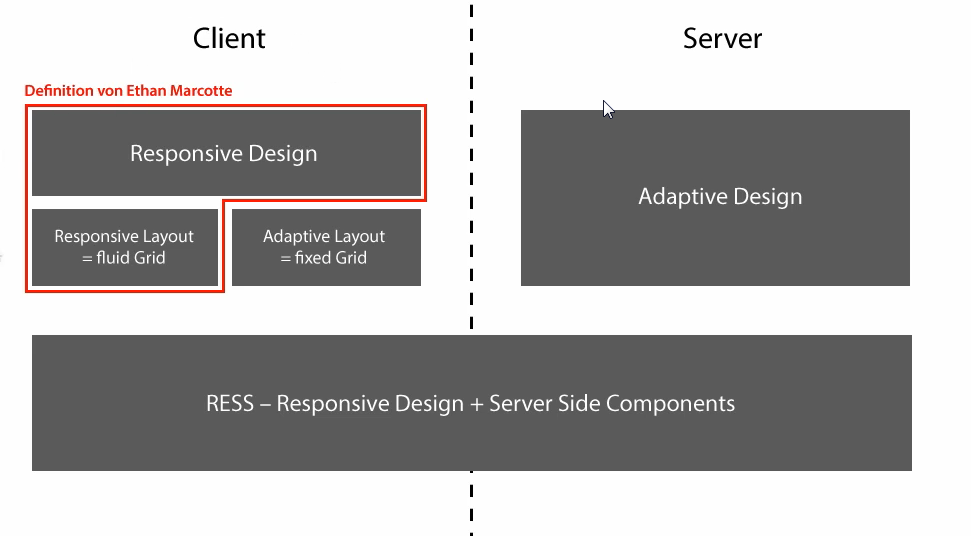
Verschiedene Layouttypen  
Static, fluid und elastic  
Fluid hat Ober- und Untergrenze, Planung ist komplexer, die Elemente werden in den Proportionen auch verändert.Oft werden alle 3 Layouts kombiniert, fluid und elastic sind echte responsive Techniken.

Es gibt für die verschiedenen Größen Breakpoints.

Man kann im Link verschiedene Medientypen angeben …   
Man kann @import ein zweites Stylesheet in ein anderes CSS einbinden …

In der CSS-Datei @media print für Druckausgabe … Hier kann ich mit „display:none; „ bestimmte Element ausblenden/umformatieren …  
media queries können verschiedene Möglichkeiten abfragen: neben and(min/max-width) gibt es and (orientation:landscape/portrait)

Responsives Layout im Gegensatz zu adaptivem formatiert für alle Größen = keine Platzverschwendung. Es steht das Design und Inhalt im Vordergrund, nicht das Gerät. Nachteil: komplexere Planung und Gestaltung, Pflege für den Kunden anspruchsvoller.



Auf dem Server ist letztlich das adaptive Layout gespeichert, beim Client wird dann möglichst nach Ethan Marcotte dargestellt.  
RESS ist dann die Hybridlösung

In der Regel wird heutzutage mobile first entwickelt. Beim Inhalt spricht man von „content first“ …  
Bei mobile-first wird weniger Code benötigt, weil kleiner Größe ohnehin als Standard 100%-Formatierungen haben, daher muss man hier nicht sonderlich viel am Anfang formatieren.  
Man braucht deutlich weniger Code und es lässt sich viel schneller entwickeln.  
Da bis IE 8 keine Media-Querys verstanden werden, sollte ggf. mit Respond.js nachgerüstet werden, es werden hiermit aber nur die media-queries min-und max-width geladen (was ja in der Regel reicht).

Für ein Touch-Screen sollten es für Schaltflächen mindestens 44px eingeräumt werden.  
Slider sind wichtig, Pfeile rechts und links reichen nicht.  
Mouseover-Effekte sind nicht mehr sinnvoll, da es natürlich auf Touchs ankommt.

In der Chrome-Entwickler-Ansicht kann man gut direkt in der Ansicht entwickeln, dann nur noch kopieren und in den Quelltext einfügen.

[www.caniuse.com](http://www.caniuse.com) zur Sicherstellung von Browserkompatibiltäten, [www.simevidas.jsbin.com/gufoko/quite](http://www.simevidas.jsbin.com/gufoko/quite) als Möglichkeit eines Autoprefixers, aber letztlich ist   
[www.modernizr.com](http://www.modernizr.com) Standard, der Klassen und Funktionen abprüft; damit ggf. png gegen svg ausgetauscht wird, svg ist hochauflösend  
<https://www.webdesignrankings.com/resources/rqrwd/>