**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.

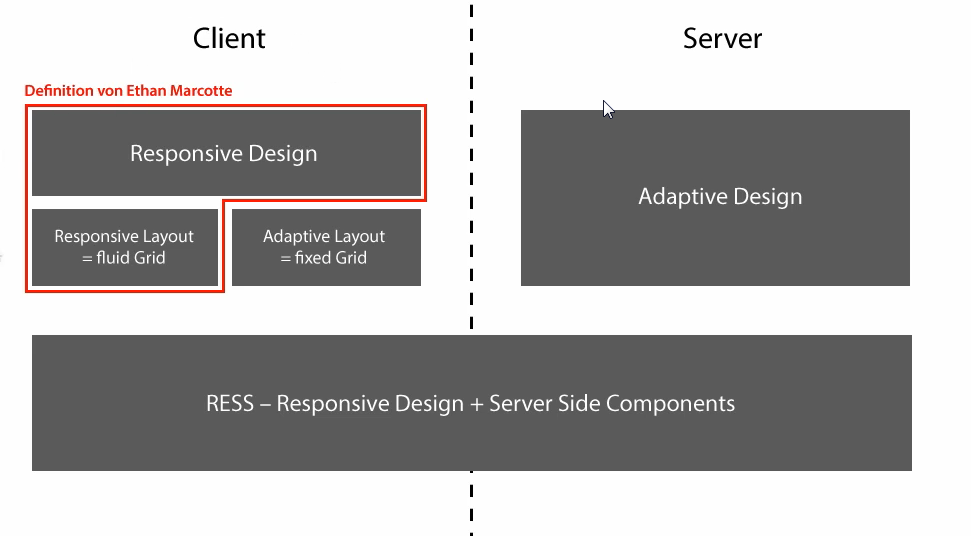
Verschiedene Layouttypen  
Static, fluid und elastic  
Fluid hat Ober- und Untergrenze, Planung ist komplexer, die Elemente werden in den Proportionen auch verändert.Oft werden alle 3 Layouts kombiniert, fluid und elastic sind echte responsive Techniken.

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/>

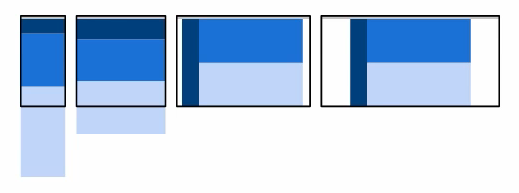
In der CSS kann man mit   
#element{  
 width:calc(100% - 440px -50px);   
}  
einem Element eine Weite zuweisen, die jene eine anderen Elementes abzieht.

10.09.18  
Ab jetzt Dozent Michael Hassel  
  
Wenn man Container positioniert, dann kann man in der Reihenfolge bei Veränderung responsive strukturieren, man kann dann aber auch (mit einer Klasse) ein float:right festlegen, sodass verschiedene Elemente nacheinander nach rechts gefloatet werden.  
  
**Mostly fluid**

Bei Google beschrieben, oder aber auch Responsive Patterns bei [https://github.com/bradfrost/this-is-responsive/](https://github.com/bradfrost/this-is-responsive/patterns.html)

Unterelemente kann man in css über nth-of-type ansprechen, z.B.  
.className:nth-of-type(3)

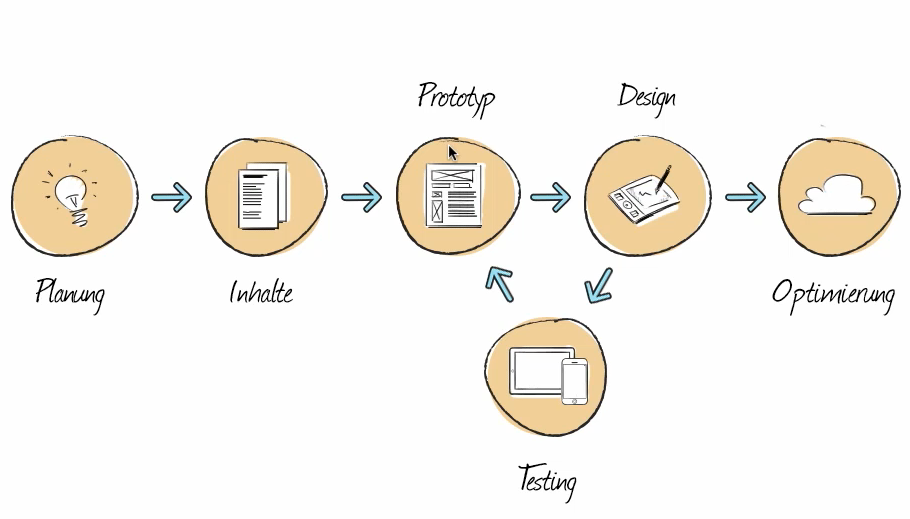
Column-Drop-Layout:  
Es rücken nach und nach die Spalten, die in der Desktop-Ansicht nebeneinander liegen, untereinander …

Layout-Shifter   
  
  
  
Kacheldesign

Je nach Anzahl der Kacheln width:x%; padding:0 0 x% 0;

Fixieren des Footers am unteren Rand. Container min-height:100% bewirkt, dass der Inhaltsbereich immer so hoch ist wie die Webseite, solange der Inhalt nicht länger/höher ist.  
  
Mit vh kann ich in der CSS eine viewport-height festlegen, bei 100% ist das immer die jeweilige Browserhöhe, vw gibt eine Bildschirmbreite …

Planung eines Webauftritts bei responsivem Design (Feedbackphase viel länger und auch nach Seitenerstellung i.d.R. öfter)



**Frameworks/Prototypen**

Vorlagen über   
<https://foundation.zurb.com/prototyping.html>

Prototypen am besten so abgespeckt und funktional/häßlich wie möglich entwerfen, damit Kunde weiß, dass es noch viel grafische Arbeit gibt.  
<http://demo.protosharesites.com/dr-flue-mobile/Photos/index.html>  
<http://getbootstrap.com/docs/4.1/examples/>

Entweder selbstentwickeln oder auf Codevorlagen und Pattern zurückgreifen  
<http://bradfrost.github.io/this-is-responsive/patterns.html>

Speziell für Navigationen  
[www.responsivenavigation.net/index.html](http://www.responsivenavigation.net/index.html)

Arbeiten mit Frameworks

Ebenfalls über bootstrap oder foundation

<http://getbootstrap.com/docs/4.1/components/alerts/>

Prototyping-Tools (hier wird aber kein Quellcode angewandt, sondern muss im Nachhinein entwickelt waren. Aber zur Konzeption optimal,   
[www.balsamiq.com](http://www.balsamiq.com)  
[www.axure.com](http://www.axure.com)  
[www.creative.adobe.com](http://www.creative.adobe.com)

**Design-Module**

Seiten sind über Module aufgebaut. Alle Module zusammen nennen sich Content Inventory.  
(idealerweise sollten Module mit gleicher Funktion auch gleich oder wiedererkennbar ähnlich aussehen)

Es gibt Ansätze zur Gestaltung, z.B. bradfrosts Atomic Design. Er geht von atomaren Strukturen aus, die sich dann aufeinander aufbauen.  
Es gibt die Schritte  
Atom-Molekül-Organismus-Template-Seite  
<http://demo.patternlab.io/>  
Hier kann man die einzelnen Schritte anschauen und ggf. zusammenstellen.

**Styles/Design**

Über   
[www.bootswatch.com](http://www.bootswatch.com)  
kann man auch online Looks der einzelnen Elemente anpassen  
Bootstrap Standarddesign  
<https://bootswatch.com/default/>

Über  
<http://styletil.es/>  
<http://samanthatoy.com/washington-examiner/>

kann man verschiedene Style optimieren und entwickeln. Hier kann man den Look einzelner Module entwickeln. Man kann hier verschiedene Stilvariationen ausarbeiten.

Andere Styleguides:  
<https://www.starbucks.com/static/reference/styleguide/>  
<https://www.mozilla.org/en-US/styleguide/>  
<https://design.firefox.com/>

**Breakpoints**

Design-Größen:  
320px IPhone4  
767px IPad hoch  
1023px IPad Quer

Da die Information im Mittelpunkt steht, wird umgebrochen, wenn das Design auf dem Gerät einen Breakpoint braucht.  
Breakpoints niemals aufgrund von Geräten erstellen (es gibt ohnehin mittlerweile viel zu viele)  
  
**Testing**

Am PC kann man hochauflösende Displays schlecht simulieren. Auch der Betrachtungsabstand ist nicht simulierbar.  
Auf echten Geräten testen: parallel über synchronisieren von Inhalten über Edge Inspect CC (Adobe)  
<https://creative.adobe.com/de/products/inspect>  
Die Erweiterung muss dann bei Chrome als PlugIn eingerichtet werden. Es muss dann eine App auf dem mobilen Endgerät eingerichtet werden …  
Man muss dann im gleichen WLAN die Seite entwickeln und es wird dann sogleich synchronisiert.  
  
Es gibt weiterhin   
[www.vanamco.com/ghostlab](http://www.vanamco.com/ghostlab)  
Dieses Tool braucht keinerlei App auf einem Endgerät. Man kann dann aber jegliche Webseite mit einer index.html/.php testen. V.a. kann man verschiedene Browser testen!!!!  
  
**Raster**  
  
Nicht-Raster-geordnete Layouts werden unruhiger, vom Gehirn anstrengender und unprofessioneller von der Gestaltung her wahrgenommen.

CSS-Grid-System

Für jede mögliche Spaltenbreite stellt das CSS-Grid eine Klasse zur Verfügung. Bestimmte Elemente erhalten dann keine eigene Breite mehr, sondern ihnen wird eine Rasterbreite zugewiesen, was zu einem einheitlicheren Layout führt.  
Demonstration des Gridsystems auf  
<https://960.gs/>  
Hier gibt es ein 12- und ein 16-spaltiges Grid-System.

Das 960.gs ist nicht mehr ganz aktuell, weil es immer noch mit px arbeitet statt mit %-Werten.  
Im responsiven Design wird mit verschiedenen Layouts gearbeitet, die bei diversen Breakpoints unterschiedliche Raster zur Verfügung stellt.

Feststellung der Seitenbreite in JS  
var \_cWidth = element.width();

Beaver ggf. als Alternative zu HeidiSQL. Und MariaDB entwickelt eigene SQL-Varianten, die ggf. in MySQL nicht drin sind ….

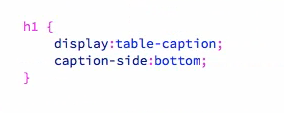
Mit box-sizing:border-box; werden die border der Container nach außen gerechnet und die margin/paddings nach innen angepasst.

**Content Choreography**  
Trennung von HTML-Code und Darstellung

Wenn HTML-Code an einer anderen Stelle steht wie die Darstellung es vermuten lässt, kann man SEO effektiver gestalten, d.h. relevante Inhalte an den Anfang, irrelevante ans Ende.

Zum Umdrehen kann man die Anweisung   
display:table;

nutzen, die dann das Element als Tabelle behandelt.  
Den Unterelementen kann man nun ein



Zuweisen, das dann das Element ans Ende des Headers steht.

**Flex-Box**<https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

Im Prinzip sind float-Eigenschaften zum Textumfluss gedacht gewesen.  
Daher gibt es mittlerweile als Standard flex-box.  
Es wird mit  
display:flex;  
deklariert.  
Die Standardrichtung ist row und zwar horizontal von links nach rechts.

Zur Positionierung gibt es die Horizontale und die Vertikale:   
justify-content:flex-start/center/flex-end;  
Um Spalten zu positionieren:  
flex-direction:column;

Um die Flexbox-Reihenfolge zu ändern, kann man Container anders sortieren:  
display:flex;  
flex-direction:column; (Sortierung der einzelnen Container.  
Innerhalb des Containers kann man nun mit   
section:nth-of-type(x){  
 order:1;  
}

section:nth-of-type(y){  
 order:2;  
}

Zum Üben:  
<http://the-echoplex.net/flexyboxes/>  
Framework:  
<http://flexboxgrid.com/>  
Eine Sammlung der Bibiliotheken mitsamt Modernize ist html5-Boilerplate.  
  
Ein interessanter Online-Seitengenerator ist   
<http://sanusart.github.io/html-now/>  
auf dem man sich den Aufbau eigener Seiten zusammenklicken kann.

Eine Möglichkeit, zwischen Bootstrap, Responsive und Boilerplate zu wählen:  
[www.initializr.com](http://www.initializr.com)

<http://goldengridsystem.com/>  
als Gridsystem-Vorlage

Für Suchmaschinen ist die robot.txt wichtig.  
  
Ein Framework  
<https://gumbyframework.com/>

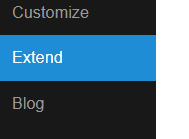
Was ist wichtig bei Frameworks:  
- Aktualität  
- ist der Inhalt/Struktur präsentiert  
- Einarbeitungszeit  
- Popularität (ggf. lohnt sich längere Einarbeitungszeit)  
- SASS/LESS

Sammlung von Frameworks   
<http://best-web-design-tools.com/>  
Seite des Autors

Vergleich von Frameworks  
<http://usablica.github.io/front-end-frameworks/compare.html>

Vergleich von Foundation und Bootstrap  
<http://designmodo.com/bootstrap-vs-foundation>  
Die Verwendung von Bootstrap hat die Gefahr, dass man sich den Einheitslook reinholt, Foundation setzt explizit auf weniger vorgestaltete Elemente.

Es gibt weiterhin   
<http://purecss.io>  
Es gibt hier einen Skinbuilder, in dem man Elemente gestalten kann.  
<http://yui.github.io/skinbuilder/>

Über „Extend“ kann man andere z.B. Bootstrap-Module reinladen.

SASS Plugin für Netbeans …!!!

Platzhalter einbauen in Prototypen über:  
<http://placehold.it/100x100/>   
<img src=“<http://placehold.it/100x100/>“ alt=““>

**Edge Reflow**

Wird von Adobe nicht weiter unterstützt

Ein weitere Editor ist

[www.froont.com](http://www.froont.com)  
hier muss man sich auch anmelden …

**Typographie**

Seiten: Typekit, Edgewebfonts, Shopfonts  
Umgewandelte Webfonts fürs Internet sind wichtig.  
Ausreichende Art von Schriftschnitten

Falsche Schriftschnitte sind no-go  
Auf Rendering und Darstellung achten.

Schriftfonts früh testen, am besten mit WindowsXP und hier den „ClearFont“ ausschalten; wenn die Schriften dann noch lesbar sind, dann ist alles gut.

Auf Farben achten … Schwarz auf Weiß ist oft ungut, weil bei hellem Licht auf nichtentspiegeltem Display es oft unerkennbar wird, da der Kontrast zu hoch wird.

<https://github.com/LeaVerou/contrast-ratio>

Schriftgrößen statt px lieber em oder rem angeben.  
em bezieht sich als Standard auf 16px, soweit nichts anderes angegeben ist.  
Zeilenhöhe line-height bezieht sich dann immer auf das jeweilige Element, d.h. em muss dann im Verhältnis zur font-size angegeben werden.  
Wenn also 0.875em angegeben sind, dann ist der line-height-em-Wert, wenn er 20px betragen soll,  
0.875\*16/20 !!!!  
max-width etc. ist ebenfalls so zu errechnen.   
rem (root em)

Man definiert im Root-Element, also html, die px-Einheit als font-size:10px;

Dann kann man in allen Elementen z.B. für 16px 1.6rem nehmen usw. .. Es werden die Abhängigkeiten bei line-height etc. völlig vernachlässigt.  
So muss man in der @media-query lediglich in der

html{  
 font-size:16px;  
}

verändern und schon ist alles auch in den Verhältnisangaben angepasst.

Wichtig ist es auch, darauf zu achten, dass sich die Schriftgrößen den Bildschirmen anpassen und nicht bei den Breakpoints unangepasst bleiben.  
Optimale Zeilenlänge sind 75 – 80 Zeichen pro row, dann Zeilenumbruch.  
Dann in der

@media screen and (min-width:xxxpx){  
 html{  
 font-size:xxpx;  
 }  
}

Wenn der Text nicht mehr größer warden soll, dann im Element mit   
max-width:xxxpx;   
begrenzen

Über css-Tricks kann man in  
<https://css-tricks.com/bookmarklet-colorize-text-45-75-characters-line-length-testing/>  
die Zeilenlänge im Browserfenster (Lesezeichen speichern) dann darstellen lassen.

**Silbentrennung**

Wird in Chrome nicht unterstützt.

body{  
 hyphens:auto;  
 -moz-hyphens:auto;  
 -webkit-hyphens:auto;   
}

Es benötigt unbedingt im html eine lang-Angabe

html{  
 lang=“de“;  
}

Weiterhin, um zu lange Zeichenketten umzubrechen

body{  
 word-wrap: break-word;  
}

Mit den HTML-Tags  
<wbr> kann man bei Bedarf einen Umbruch erzwingen, &shy; erzwingt bei Bedarf ein Trennzeichen.

Es ist auch möglich, die Schrift proportional zur Displaygröße zu skalieren.  
[www.trentwalton.com](http://www.trentwalton.com)  
Man kann hier die Einheiten vh und vw angeben.  
Sinnvoll ist, als Fallback-Möglichkeit zunächst rem anzugeben, dann vh, sodass bei Inkombatibilität rem verwendet wird.

Umwandlung einer Printschriftart in einen Webfont mittels   
[www.fontsquirrel.com/tools/webfont-generator](http://www.fontsquirrel.com/tools/webfont-generator)  
Man kann auch entweder über Adobe oder Google webfonts einbinden:  
Hier eine Schriftart auswählen, dann über <script></script> den angebotenen Link zum Font im Head einbinden und schließlich im Element über font-family den Font laden.

Über font-awesome lassen sich zudem Icons einbinden.  
Es müsste aber hier ein recht großes Iconset eingebunden werden.

Über <https://icomoon.io/app/#/select/library>  
lässt sich ein individuelles Set erstellen.

<https://www.stepstone.de/stellenangebote--Professional-Frontend-Developer-m-w-Mannheim-Stuttgart-Frankfurt-Duesseldorf-Essen-Koeln-Atos--5205933-inline.html?suid=ac1ce716-3727-4640-a35f-d25f42e9d2c4&rltr=6_6_25_dynrl_m_0_0>

**Navigation**Alle Seiten sollte auch über Links im Inhalt erreichbar sein, da es einige User gibt, die einfach scrollen, ohne z.B. den Hamburger-Button nicht zu benutzen, um die Navigation auszuklappen.

Es sollten „Do-Nothing“-Navigationen benutzt werden, also solche, die vom Nutzer überhaupt keine Aktion verlangen. Man braucht zudem immer Fallback-Lösungen, da es auch User gibt, die JS ausgeschaltet haben.  
Man kann es über eine Select-Box lösen (hässlich), oder über Slider

*Off-Canvas-Navigation*

Navigation sollte auf Smartphones nicht größer als 280px sein.

[www.berriart.com/sidr](http://www.berriart.com/sidr)

Es gibt Portalseiten, d.h. Seiten mit untergeordneten Links, man lässt dann in der mobilen Navigation Unterlinks/Dropdowns weg. (Macht Sinn, wenn die Navi zu lang wird)

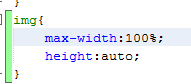
**Flexible Grafikelemente**

HTML-Grafik oder CSS

Wenn Bilder inhaltlich irrelevant sind, dann background-image, wenn relevant, dann <img>  
<img>s lassen sich via Drag&Drop wegziehen, backgrounds nicht.  
<img>s lassen sich besser proportionieren, css-backgrounds lassen sich besser austauschen.  
background-size:contain; das Bild wird immer im Container vollständig dargestellt, background-size:cover; Der Container wird immer ausgefüllt, das Bild aber beschnitten.  
Suchmaschinen ignorieren background-Bilder  
  
Prototyping-Programm „Reflow“

Ein Online-Tool zum Gestalten von UML, Datenbank-Darstellungen, Grafiken u.v.m., speichert in XML ab …  
<http://draw.io>

Formatieren eines img-Elements:



*Fokuspunkte definieren (zum Zommen)*

Mögliche Quellen für eine focal-point-Platzierung:

<https://github.com/adamdbradley/focal-point>

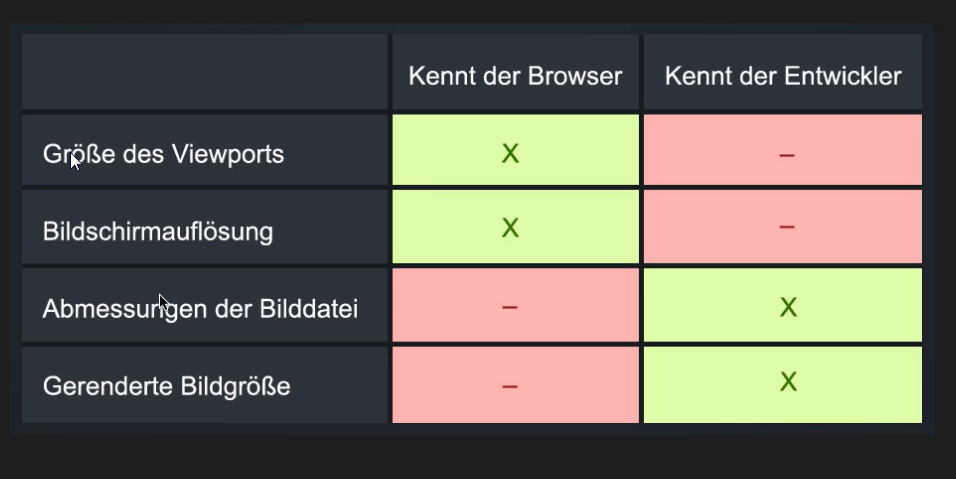
[www.designshack.net/articles/css/focal-point-intelligent-cropping-of-responsive-images/](http://www.designshack.net/articles/css/focal-point-intelligent-cropping-of-responsive-images/)

*Austausch von <img> über CSS*

[www.responsiveimages.org](http://www.responsiveimages.org)

Austausch aus Gründen von

* Bildschirmen, z.B. Retina-Display
* Viewports (media queries)
* Verschiedene Grafikformate
* Gestalterischem Hintergrund



Möglichkeiten in HTML zur Bildauswechslung sind Definitionen von Sourcesets und sizes, die Möglichkeiten sind kombinierbar.



Polyfills für srcset und sizes  
  
<https://scottjehl.github.io/picturefill>  
bindet für verschiedene Browser ein  
<http://adaptive-images.com/>

Firefox macht flexbox erst ab 2018 mit aktueller Version, die vorletzte ist schon nicht mehr kompatibel.

**Flexible Inhalte** *Videos*

Videointegration über <video src=“quelle.mp4“ type=“video/mp4“></video>

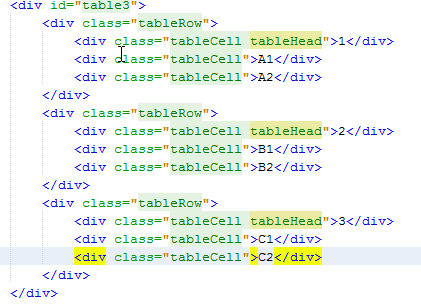
Bzw. über ein <iframe></iframe>  
Einbindung des Videos über   
<https://github.com/davatron5000/FitVids.js>

Eine Ressource zum Abrufen von JS-Codes, die man via Link dann Einbinden kann (Performance-Vorteile!!!! im Produktionsmodus)   
<https://developers.google.com/speed/libraries/devguide>

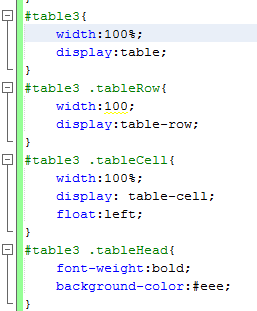
*Tabellen*

Für responsive Darstellungen mobile in CSS td, th{width:100%, float:left; }  
  
*Tabellen nachbauen mit CSS*

HTML



CSS



*Image-Maps*

JS-Bibliothek über   
<https://github.com/stowball/jQuery-rwdImageMaps>  
(Technik genauer anschauen ggf., vllt. auch neuer Technik … Ist wohl veraltet)

*Slider-Integration*

JS-Bibliothek über   
<https://kenwheeler.github.io/slick>  
Hier findet man unter „settings“ die einzelnen Einstellungen …

Tool zum Simulieren von Smartphones (z.B. beim Antippen eines Formulars Tastaturanzeige)  
<http://responsive.victorcoulon.fr>  
  
Formulare: Placeholder bei <input> nur als Beispiel, nicht als Ersatz einer Feldbezeichnung benutzen …  
<input type=“url“>  
Auf Smartphones ändern sich dann die automatischen Tastaturausgaben bei anderen types  
<input type=“date“> ruft einen Datepicker auf.

Für Smartphones ist aber auch möglich:  
<a href=<mailto:email@somewhere.de?subject=Bemerkung%20zum%20Versenden>>Email schreiben</a>

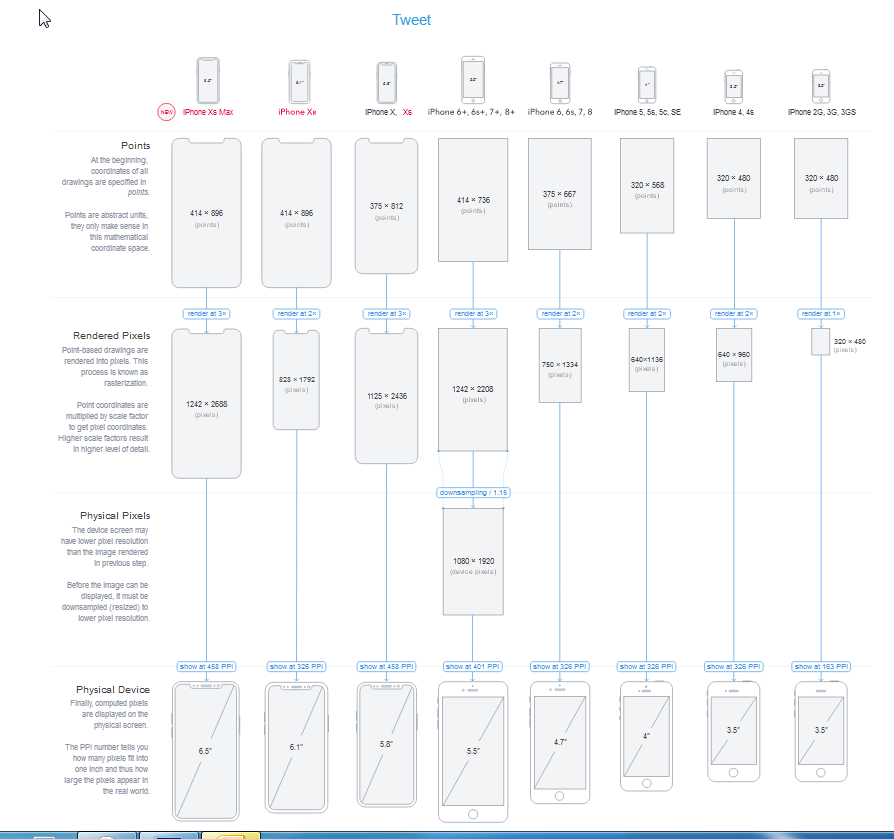
Anklickbares Telefon über  
<a href=“tel:+498154711“>Anruf</a>

Anklickbare SMS  
<a sms=“tel:+498154711“>SMS senden</a>

**Hochauflösende Displays**

Aufführung der Display-Varianten auf  
[www.paintcodeapp.com/news/ultimate-guide-to-iphone-resolutions](http://www.paintcodeapp.com/news/ultimate-guide-to-iphone-resolutions)

Hier werden die Multiplikatoren erläutert  
Normaler Bildschirm x1, IPhone4 2x usw.



[www.screensiz.es/phone](http://www.screensiz.es/phone)

zeigt verschieden Weiten und Pixeldichten der unterschiedlichen Geräte an. Her kann man die media-queries ersehen, was sehr wichtig ist und die Auflösungen.  
Man kann letztlich Grafiken immer x-mal so groß entwerfen und dann in einen Div-Container mit einfachem Ausmaß einfügen. Für doppelt so große Bilder gibt es die Synthax [bild@2x.jpg](mailto:bild@2x.jpg) usw..

Über den media-query  
@media screen and (-webkit-min-device-pixel-ratio:2),  
screen and (min-resolutein:192dpi),  
 screen and (min-resolutein:2dppx){

.divDatei{  
 background:url(‚pfad/bild@2x.jpg‘) 0 0 no-repeat;  
 background-size:100% 100%; //bzw. Pixelangaben, die benötigt werden

}

}

Eine Testseite für das aktuelle Gerät ist

[www.bjango.com/articles/min-device-pixel-ratio](http://www.bjango.com/articles/min-device-pixel-ratio)

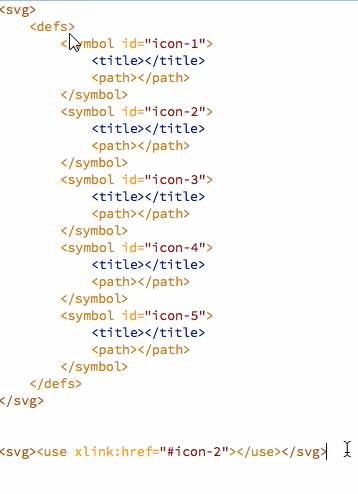
Bei der Auswahl von verschiedenen Größen im HTML-Format machen nur begrenzte neuere Browser mit, es gibt hierzu ein Workaround auf   
<http://imulus.github.io/retinajs>

Hier muss man die verschiedenen Bilder in einer gleichen Datei ablegen, je nach Notwendigkeit holt sich das Skript die entsprechende Datei von selbst.  
[www.dahliacreative.com/retinisejs](http://www.dahliacreative.com/retinisejs)  
bietet das Gleiche auf Basis von JQuery …

SVG (Vektorgrafik) ist für hochauflösend und skalierbar optimal.  
  
Quelle für SVGs:  
<https://google.github.io/material-design-icons>  
Tool zur Arbeit mit SVG-Icons  
[www.grumpicon.com](http://www.grumpicon.com)  
 **Geschwindigkeit optimieren**

Chrome-AddOn   
[www.yslow.org](http://www.yslow.org)  
  
http-Requests sind entscheidend für die Geschwindigkeit des Browsers  
Über network in den Entwicklertools sieht man die http-Requests  
Optimierung von JS- sowie CSS-Dateien am Entwicklungsende macht Sinn.  
[www.cssshrink.com](http://www.cssshrink.com)  
Mehrere Dateien kann man über   
[www.shrinker.ch](http://www.shrinker.ch)  
verkleinern.

Ansonsten immer möglichst auf externe Quellen zurückgreifen.  
Weiterhin kann man Sprites verwenden. CSS- und SVG-Sprites:

Um Koordinaten und Größen verschiedener Spriteelemente zu ermitteln, gibt es   
[www.spritecow.com](http://www.spritecow.com)  
Da png-Grafiken nicht über CSS einfärbbar sind, die Größen-Optimierung bei responisven Seiten oft Probleme bereitet, da via px positioniert wird und die Skalierung nicht möglich ist und auch die Pflege schwierig ist, werden i.d.R. SVG-Sprites eingesetzt.  
2 Arten SVG-Sprites  
a) SVG-Sprite-Paket mit verschiedenen Spriteseiten über   
<https://github.com/google/material-design-icons>  
Einbindung wie üblich über Koordinaten  
b) Der SVG-Code wird direkt im Dokument platziert (wesentlich praktischer), wird als „Spritemap“ bezeichnet  
Man kann es recht umständlich mit   


Einbinden, oder aber über Icomoon  
<https://icomoon.io> (s. auch oben)  
ein Spritesheet erstellen lassen   
  
Eine skalierbare SVG einbinden  
<https://www.mediaevent.de/tutorial/svg-img-sprite.html>

**Grafiken optimieren**gifs nur noch bei Animationen, es gibt neuerdings auch WebP, das aber nur von Chrome unterstützt wird.

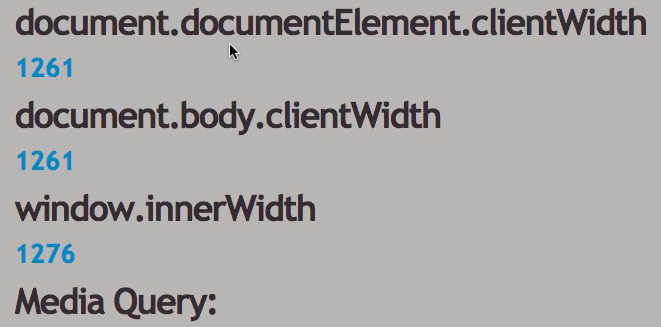
Zum Komprimieren von Dateien gibt es die Seiten   
[www.compresspng.com](http://www.compresspng.com)  
[www.compressjpg.com](http://www.compressjpg.com)  
auf denen man online die Bilddateien optimieren kann.  
Um weiter zu optimieren, kann man die Datei auch in HTML-Code umwandeln  
[www.askapache.com/online-tools/base64-image-converter](http://www.askapache.com/online-tools/base64-image-converter)  
Die erzeugte Datei kann man letztlich über CSS oder direkt in HTML einbinden.



**JavaScript für responsibles Design***Conditional Loading*  
  
Zunächst auf kleinster Ebene anfangen. Dann Viewport-Breite abfragen und dann komplexere Skripte ausführen.  
Die Skripte werden nur einmal geladen, werden dann aber nicht nachgeladen.

JS-Abfrage der Viewportgröße über   
document.documentElement.clientWidth  
 *media-queries in JS*

matchMedia, man kann hier verschiedene Mediaqueries festlegen, funktioniert aber nur nach Veränderung  
  
Verschiedene Width-Abfragemöglichkeiten:

  
mit unterschiedlichen Werten  
Media Query entspricht window.innerWidth, die anderen Werte weichen von Media Query um 15 px ab. Wenn also eine Kombination mit Media Queries, dann mit window.innerWidth und matchMedia!

Hilfreiche Skripte:  
auf Basis von resize:  
[www.simplestatemanager.com](http://www.simplestatemanager.com)  
kann Smartphone, Tablet, PC simulieren

matchMedia  
<http://wicky.nillia.ms/enquire.js/>

Polyfill für matchMedia  
<https://github.com/paulirish/matchMedia.js>  
  
Kerntechniken werden vorgestellt auf  
[www.responsivejavascript.com/#Libraries](http://www.responsivejavascript.com/#Libraries)

Abschließend:  
Prinzipiell Responsibilität versuchen, mit CSS zu lösen (Performance), wenn es nicht reicht, dann JS anwenden.