RESPONSIVE DESIGN

PROF. ANDREAS HARTMANN - WEBTECHNOLOGIEN

THEMEN

- Layouttypen, flexible Layout
- Media Queries
- Zugänglichkeit
- Breakpoints
- Layout Patterns
- Navigation
- Flexible Elemente
- Responsive Design testen

QUELLEN, LITERATUR

- Quellenangaben
  - Der Inhalt richtet sich – soweit nicht anders gekennzeichnet – aus an:

  [1] Responsive Webdesign, Andrea Ertel, Kai Laborenz, Galileo Press, Bonn, 2015 (http://d-nb.info/1054672083)
- Es handelt sich um eigene Abbildungen, wenn keine Quelle angegeben ist.
- Einige Inhalte sind an Wikipedia angelehnt. Als Quellangaben wurden die dort angegebenen Referenzen geprüft.

ANFORDERUNGEN AN WEBSEITEN

ANNO 2015

BEGRIFF

Ethan Marcotte –

http://alistapart.com/article/responsive-web-design

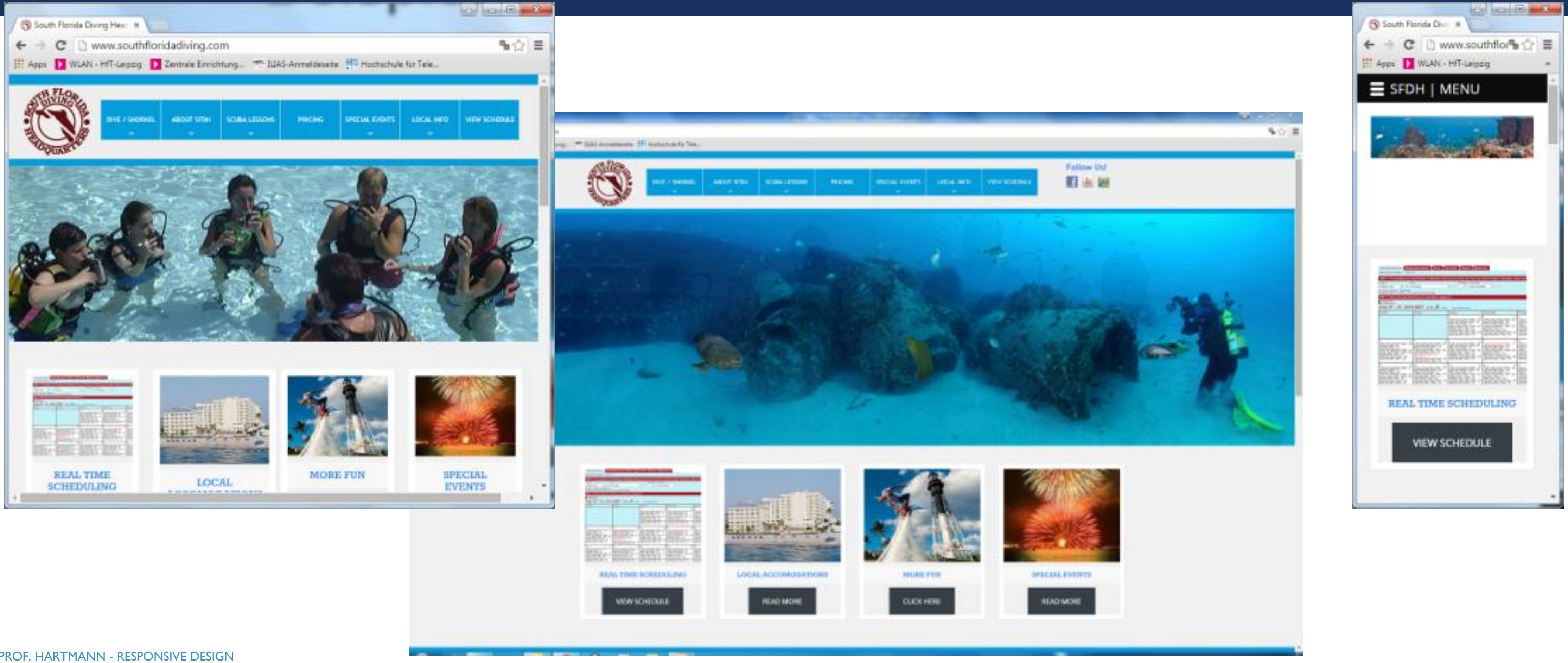
VERSTÄNDNIS

DREI ENTWICKLUNGSANSÄTZE

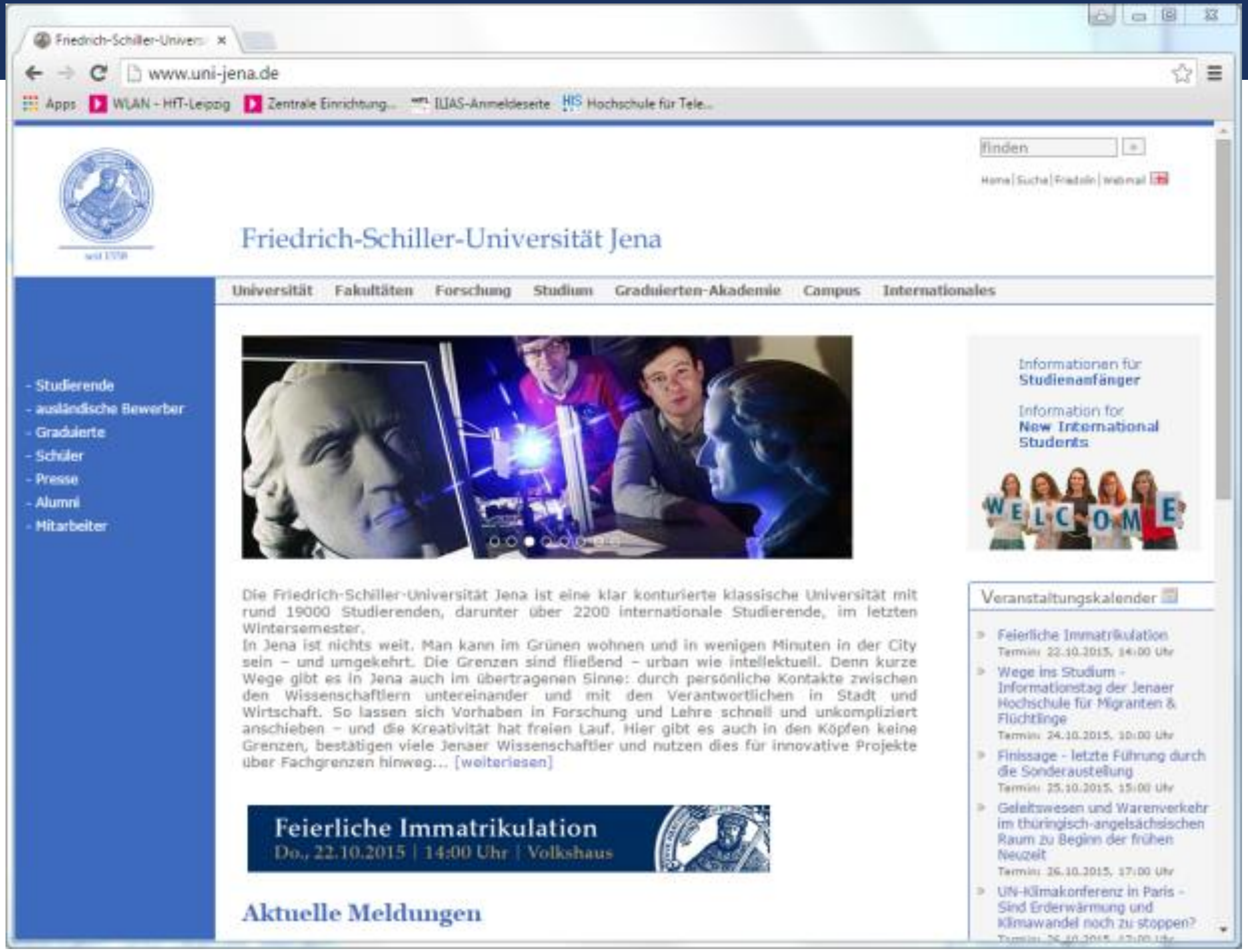
- fließendes Layoutraster
- anpassungsfähige Inhalte
- Layoutumbrüche durch Media Queries

BEISPIEL

http://www.southfloridadiving.com/



KEIN GUTES BEISPIEL…



LAYOUTTYPEN

RESPONSIVE DESIGN

ÜBERSICHT DER TYPEN

Fixe	Float	Adaptive	Responsive
• Dimensionen sind fix in Pixel angegeben • Browser reagieren mit linearer Skalierung oder Scrollbalken	• Layouts ändern sich relativ zum Größenverhältnis • Inhalte bleiben jedoch fix	• im Prinzip mehrere „fixe“ Layouts vordefiniert (CSS) • Browser wählt bestes Layout aus	• kombiniert Float und Adaptive • besitzt skalierbare Inhalte

GRID

- Für eine Webseite mit responsive Layout wird ein dynamisches und anpassungsfähiges Raster benötigt.
- Als Entwickler/Designer hilft es, in 2-dimensionalen Matrizen zu denken (Grid).
- Elemente der Matrix passen sich bzgl. Höhe und Breite dynamisch an das Browserfenster bzw. das Display an.

PRO'S & CON'S

Pro

- hohe Flexibilität für alle Bildschirmgrößen
- fließender Übergang zwischen Umbruchpunkten

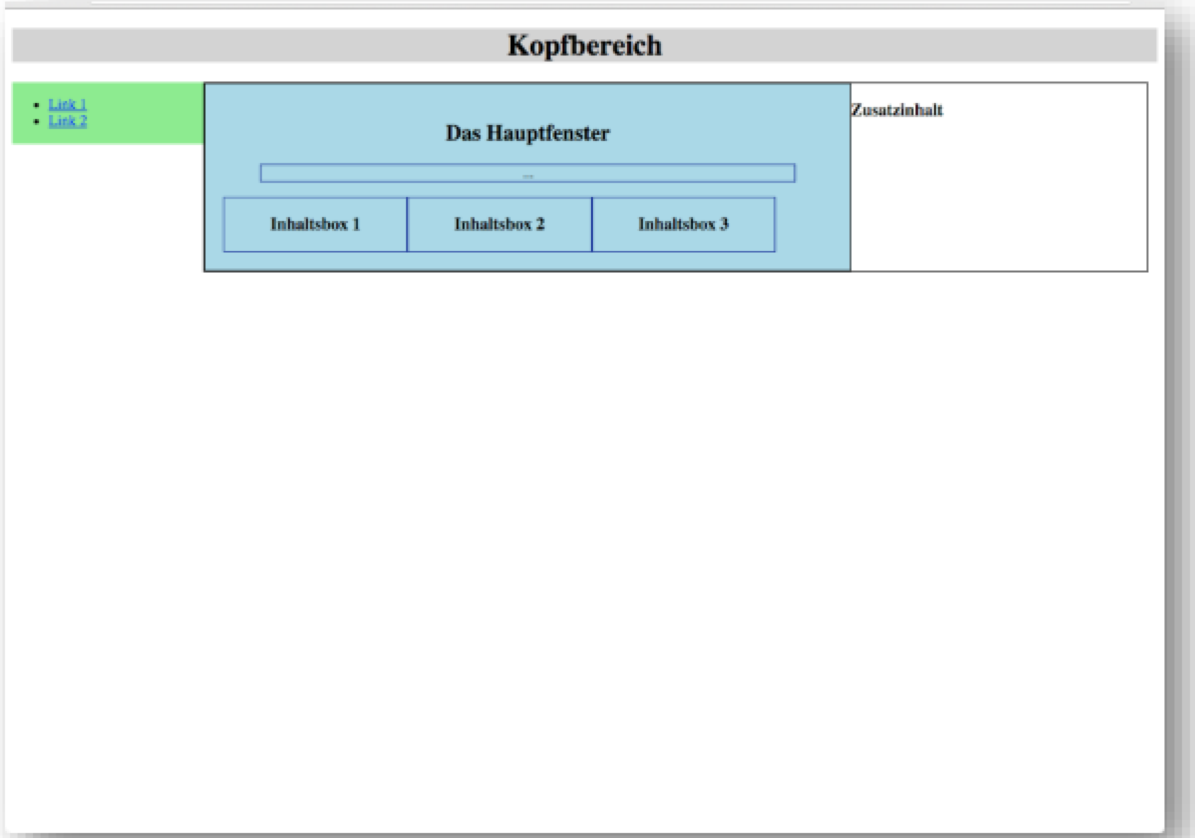
Con

- mehr Aufwand in der Entwicklung
- weniger Kontrolle auf Pixel-Ebene

FLEXIBLE LAYOUTS

RESPONSIVE DESIGN

EINE FIXE WEBSEITE



AUSZUG CSS

```
.main {
    width: 1035px;
    float: left;
    overflow: hidden;
    border: 1px solid black;
}

nav {
    width: 210px;
    float: left;
    overflow: auto;
    background: lightgreen;
}

.mainContent {
    width: 710px;
    float: left;
    padding: 20px;
    box-sizing: border-box;
    background: lightblue;
    text-align: center;
    border: 1px solid black;
}
```

Inhalte werden mit fixen Breiten gestaltet (Pixel)

Die Anpassung an diverse Anzeigegeräte ist mühsam!

SCHRITT 1 - UMRECHNUNG

- Für anpassungsfähige Layouts müssen absolute Angaben (Pixel) in relative Angaben (Prozent) umgerechnet werden
- Aufpassen bei Rundungsfehlern!

$$\text{Breite in Prozent} = \frac{\text{Breite in Pixel x 100}}{\text{Breite des Elternelements in Pixel (Kontext)}}$$

Beispiel: .mainContent hat eine Breite von 68,59903382%

SCHRITT 2 –
INHALTE ANPASSEN

- Bei Text ist nicht viel zu tun. Was ist mit Blockelementen?
- Bilder:

```
<img src=„beispiel.png“>
<img src=„beispiel.png“ width=„200“>
```

- kann mittels CSS überschrieben werden

```
img {
    width: 100px;
}
```

immer noch fix

BILDER ANPASSEN

```
img {
   max-width: 100%;
   height: auto;
}

//Alternative: Beschneiden
HTML
<figure>
 <img src=„beispiel.png“>
</figure>
CSS
figure {
 overflow: hidden;
}
```

Jetzt wird das Bild auf maximal Originalgröße und ansonsten proportional angezeigt.

Die untere Alternative beschneidet das Bild in einer Figure-Caption.

SCHRITT 3 – MEDIA QUERIES

Die bisherigen Anpassungen haben lediglich die Spalten flexibel gemacht. Das reicht jedoch nicht aus.

z.B. Smartphone-Tablet-Desktop

Mit Media Queries können Geräte- oder Browserspezifische Eigenschaften abgefragt werden, für die dann ein jeweils konkreter CSS-Code angewendet wird.

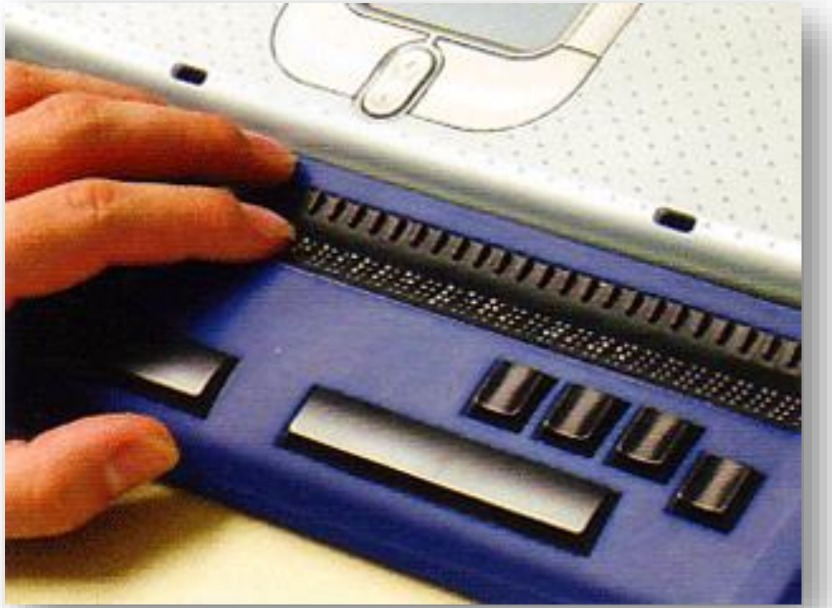
MEDIA QUERIES

RESPONSIVE DESIGN

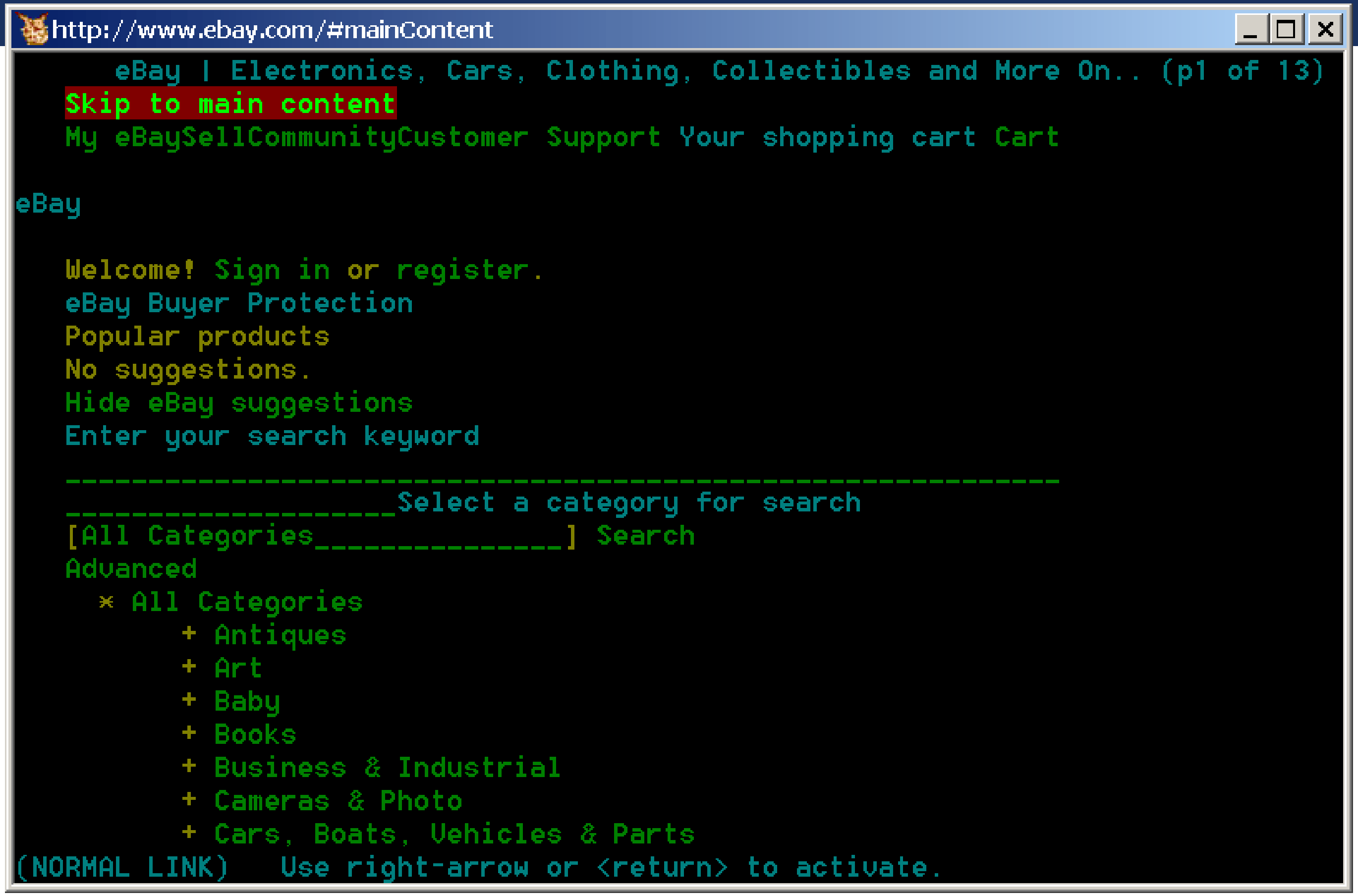
MEDIENTYPEN

media type	Ausgabeart (Anzeigegerät)
all	gilt für alle Ausgabegeräte
braille	blindenschriftfähige Ausgabe
embossed	blindenschriftfähige Drucker
handheld	ehemals: Palmtops, PDAs, WinCE-Geräte
print	Drucklayout, Druckausgabe
projection	Ausgabe mit Beamern o.ä.
screen	Ausgabe auf dem (Desktop-)Bildschirm
speech	Screenreader
tty	Ausgabegeräte mit feststehenden Zeichentypen
tv	Ausgabe auf Fernsehgeräten

AUSGABE MIT BRAILLE-ZEILE



AUSGABE TTY / KONSOLE



Quelle: http://www.kompx.com/en/windows-console-applications-web-browsers.htm

ANWENDUNG VON MEDIENTYPEN

Medientypen können in HTML oder CSS adressiert werden.

HTML

```
<link rel=„stylesheet“ href=„styles.css“ media=„screen“>
<link rel=„stylesheet“ href=„print.css“ media=„print“>
```

HTML Header

```
<style>
 @media print { ... }
</style>
```

CSS

```
@media screen {
 ...
}
```

„ALTE" MEDIENTYPEN

Gerätetypen und Geräteklassen ändern sich ständig und sehr dynamisch. Es ist immer besser, ein Anzeigegerät anhand seiner Eigenschaften zu identifizieren/zu beschreiben.

MEDIA FEATURES

- Adressierte Kerneigenschaften beziehen sich auf
  - Größen (Anzeige, Display)
  - Farbtiefen
  - Ausrichtungen (landscape, portrait)

MEDIENEIGENSCHAFTEN (1)

Eigenschaft	Beschreibung	Maße
width (min/max)	Breite der sichtbaren Ausgabe	px,%,em
heigth (min/max)	Höhe der sichtbaren Ausgabe	px,%,em
device-width (min/max)	Breite des Displays des Ausgabegeräts	wie oben
device-heigth (min/max)	Höhe des Displays des Ausgabegeräts	wie oben
orientation	ergibt wahr für portrait wenn die Höhe größer als die Breite ist	portrait, landscape
aspect-ratio (min/max)	prüft das Seitenverhältnis der sichtbaren Ausgabe	Breite/Höhe
device-aspect-ratio (min/max)	prüft das Seitenverhältnis des Ausgabegeräts	Breite/Höhe

MEDIENEIGENSCHAFTEN (2)

Eigenschaft	Beschreibung	Maße
color (min/max)	prüft die Farbtiefe des Ausgabegeräts	integer
color-index (min/max)	prüft das Ausgabegerät hinsichtlich indizierter Farben	integer
monochrome (min/max)	prüft das Ausgabegerät hinsichtlich monochromer Farbtiefe	integer
resolution (min/max)	prüft die Auflösung (Pixeldichte) in DPI oder DPCM	dpi, dpcm
scan	Testet TV-Geräte auf Vollbild (progressive) oder Halbbild (interlace)	progressive, interlace
grid	Prüft die Ausgabeeigenschaft als Raster. 0 entspricht Bitmap, 1 entspricht Raster.	0, 1

MEDIA QUERIES (SYNTAX)

```
@media [not|only] type and (expression) [AND (expression)]{
 /* CSS-Anweisungsblock */
}


@media [not|only] type1, type2 and (expr1, expr2, expr3){
 /* CSS-Anweisungsblock */
}
```

- das logische *and* muss verwendet werden, um Typen und Eigenschaften zu prüfen
- es können mehrere Typen (oder Eigenschaften) gelistet werden
- das Komma wirkt wie ein logisches *or*
- *only* wirkt für ältere Browser (Block wird versteckt)

VIEWPORTS

- Was ist mit unterschiedlichen Auflösungen (z.B.Retina)?
- Der richtige Einsatz von Media Queries setzt Wissen zu Viewports und Pixelauflösungen voraus.
- Zunächst wird mit Viewport der sichtbare Bereich des Ausgabemediums bezeichnet.
    - bei Desktops: sichtbare Bereich des Browserfensters

LAYOUT-VIEWPORT

- eingeführt von Apple für das iPhone
- Desktop-Webseiten mussten auf niedrige Displaygröße verkleinert werden (320 * 480 Pixel auf dem iPhone 3)
- mit der Zoom-Funktion wieder lesbar

DEFINITON LAYOUT-VIEWPORT

„Der Layout-Viewport ist die Bezugsgröße für die Darstellung der Webseite im mobilen Browser. Prozentuale Größenangaben im CSS beziehen sich immer auf den Layout-Viewport.“

Quelle: [1]

DEFINITON GERÄTEPIXEL

„Die Gerätepixel sind die physikalischen Pixel des Displays, also die einzelnen Einheiten, aus denen der Bildschirm [inhalt] aufgebaut ist." Ihre Anzahl und ihr Verhältnis ist hardwareabhängig und daher unveränderlich.

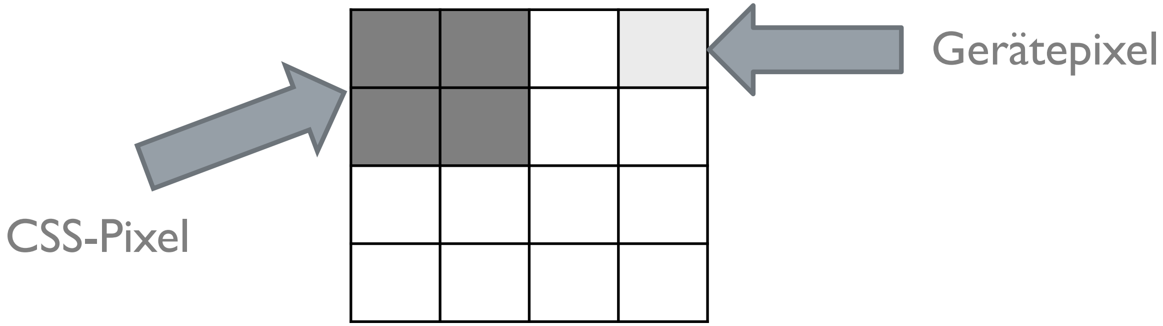
Quelle: [1]

DEFINITON CSS-PIXEL

„Die CSS-Pixel sind eine Einheit für die Darstellung der Inhalte auf dem Display. Jede CSS-Größendeklaration bezieht sich auf die CSS-Pixel.“ Die Pixeldichte gibt das Verhältnis zwischen CSS-Pixel und Gerätepixel an.

Quelle: [1]

PIXELDICHTE



Beispiel bei Pixeldichte 2

DEFINITON
VISUELLER VIEWPORT

„Der visuelle Viewport bezeichnet die sichtbare Displaygröße in Pixeln. Durch ihn blicken wir auf den Layout-Viewport der Webseite."

vgl. http://www.quirksmode.org./mobile/viewports.html

Quelle: [1]

KONTROLLFRAGE

Was bedeutet die folgende Angabe auf dem Retina-Display?

```
<meta name=„viewport“ content=„width=800px“>
```

VIEWPORT SETZEN

in HTML und CSS möglich

HTML

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

CSS

```
@viewport {
 width: device-width;
 zoom:1;
}
```

Mobile-First

- Standardlayout gilt für kleine Displays
- mit `min-width:[Breakpoint]` wird nach und nach vergrößert

Desktop-First

- Standardlayout gilt für große Displays
- mit `max-width:[Breakpoint]` wird nach und nach verkleinert

RELATIVE GRÖßENANGABE

- Bisher haben wir mit Pixeln gearbeitet (CSS-Pixel).
- Was passiert jedoch bei einem Browser-Zoom?
- Media Queries funktionieren auch hier noch – mit relativen Angaben (*em*).
- Die Einheit *em* bezieht sich immer auf die relative Schriftgröße.
- Umrechnen eines Breakpoints von *px* in *em* (Basisschriftgröße = *16 px*):

$$20\ \text{em} = \frac{320\text{px}}{16\text{px/em}}$$

PIXEL

- Physischer Pixel = festes Maß entsprechend der Auflösung des Bildschirms (vgl. native Auflösung)
- 1280x1024 Pixel ergeben 1.310.720 Bildpunkte

AUFLÖSUNGEN

dpi - Diese Einheit repräsentiert die Anzahl an Punkten pro Zoll. Ein Bildschirm hat typischerweise 72 oder 96dpi; ein gedrucktes Dokument erreicht normalerweise weit höhere dpi Zahlen. Da 1 Zoll 2.54cm entspricht, 1dpi ≈ 0.39dpcm.

dpcm - Diese Einheit repräsentiert die Anzahl an Punkten pro Zentimeter. Da 1 Zoll 2.54cm entspricht, 1dpcm ≈ 2.54dpi.

dppx - Diese Einheit repräsentiert die Anzahl an Punkten pro px Einheit. Wegen dem festen 1:96 Verhältnis von CSS in zu CSS px, ist 1dppx äquivalent zu 96dpi, dies entspricht der Standardauflösung von Bildern, die in CSS dargestellt werden, wie in image-resolution definiert.

Beispiel: `@media print and (min-resolution: 300dpi) { ... }`

Quelle: https://developer.mozilla.org/de/docs/Web/CSS/resolution

PHYSISCHE LÄNGENMAßE

Einheit	Name in CSS	Beschreibung	Beispiel
Zoll	in	Ein Zoll = 2,54 cm	margin-left: 1in;
Pica	pc	Sechste Teil eines Zolls (=12 Punkte)	font-size: 1pc;
Punkt	pt	72ste Teil eines Zolls	font-size: 12pt;
Zentimeter	cm	100ste Teil eines Meters	margin-top: 1.5cm;
Millimeter	mm	1000ste Teil eines Meters	padding: 1mm;

*vorwiegend für Ausgabemedium Druck geeignet (vgl. CSS-Pixel)

PIXEL IM VERHÄLTNIS ZU PHYSISCHEN MAßEN

Einheit	physische Größe	Größe in Pixel
1in	2,54 cm	96 Pixel
1pc	1/6 Zoll	16 Pixel
1pt	1/72 Zoll	~1,33 Pixel
1cm	1cm	~37,8 Pixel
1mm	0,1 cm	~3,78 Pixel
1px	abh. vom Gerät	0,75 Punkt

RELATIVE LÄNGENMAßE

Einheit	Name in CSS	Beschreibung	Beispiel
em	em	relative Größe mit Bezug zum Buchstaben „M" (Browser berechnet)	margin-left: 1em;
Root-em	rem	wie em, jedoch mit Bezug auf Schriftgröße im Wurzelelement des Dokuments (<html>)	font-size: 1.5rem;
ex	ex	analog zu em, jedoch mit Bezug auf den Buchstaben „x" (Schriftart)	font-size-adjust: 0.5ex;
Null-Breite	ch	ähnlich em, jedoch mit Bezug auf die Breite der Null „0" (Schriftart)	font-size-adjust: 0.5ch;
Viewport-Breite	vw	100ste Teil der Breite des Viewports	width: 25vw;
Viewport-Höhe	vh	100ste Teil der Höhe des Viewports	height: 50vh;
Viewport (min)	vmin	100ste Teil der kürzesten Seite des Viewports	max-width: 100vmin;
Viewport (max)	vmax	100ste Teil der längsten Seite des Viewports	height: 50vmax;

UNTERSTÜTZUNG BEACHTEN

Nicht alle Browser unterstützen Media Queries, daher sollte es immer ein Standardlayout geben, welches ohne diese Unterstützung klar kommt.

MEDIA QUERY
MITTELS JAVASCRIPT

- falls keine Unterstützung für Media Query vorhanden ist
- bedarf letztlich jedoch der Unterstützung von Javascript
- einige vorgefertigte Bibliotheken liefern diese Funktionalität
  - respond.js – https://github.com/scottjehl/Respond
  - css3-mediaqueries.js – http://code.google.com/p/css3-mediaqueries-js
  - Restive JS – https://github.com/obihill/restive.js

DESIGNANFORDERUNGEN

RESPONSIVE DESIGN

KEINE MAUS, SONDERN...

Heute arbeiten die meisten mobilen Geräte mit Touch-Steuerung, d.h. Bedienung mittels ein oder mehrerer Finger

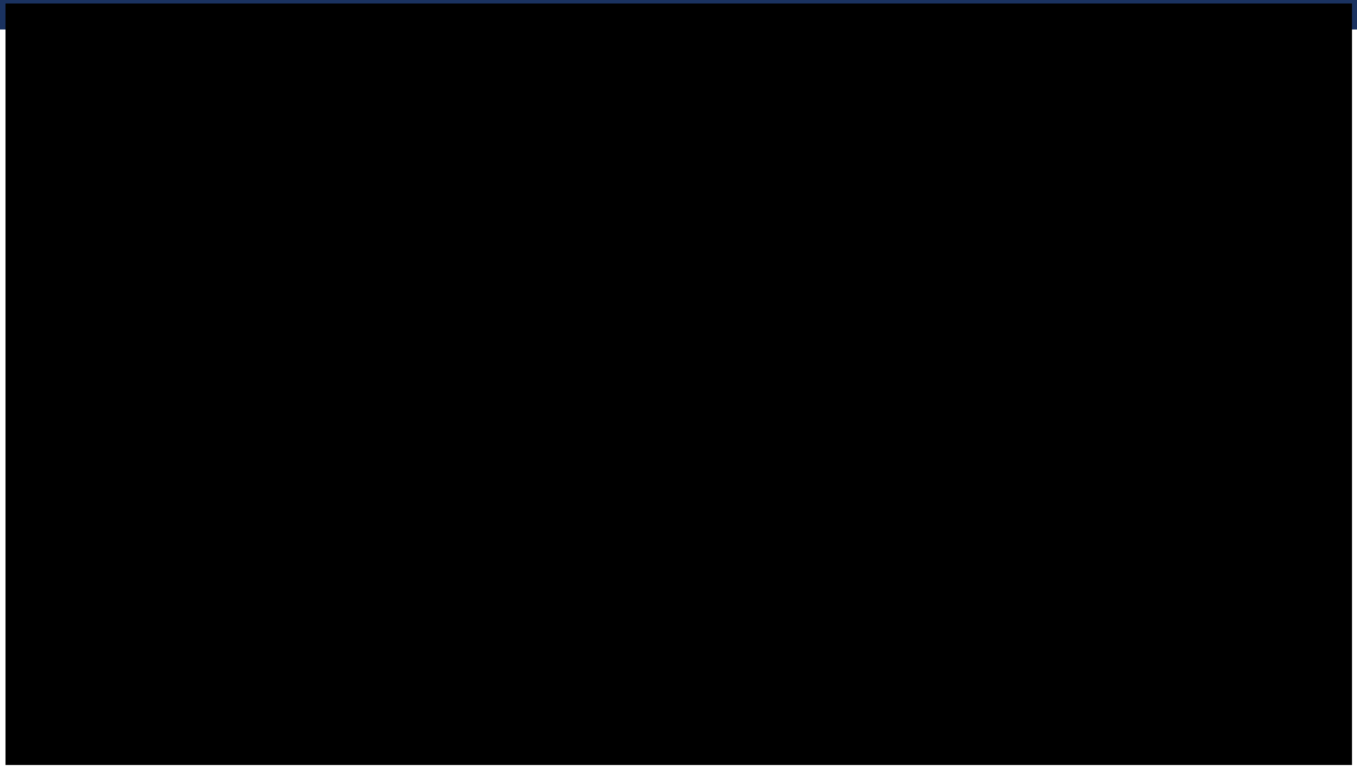
Bedienkonzepte aus der Desktopwelt basieren jedoch auf der Steuerung mit der Maus

Linksklick, Rechtsklick, ziehen mit gedrückter Maustaste...

... weicht Wischgesten, „tappen“ mit dem Finger

neue Gesten kommen hinzu (Spiel: *The Room*)

BEISPIEL: THE ROOM

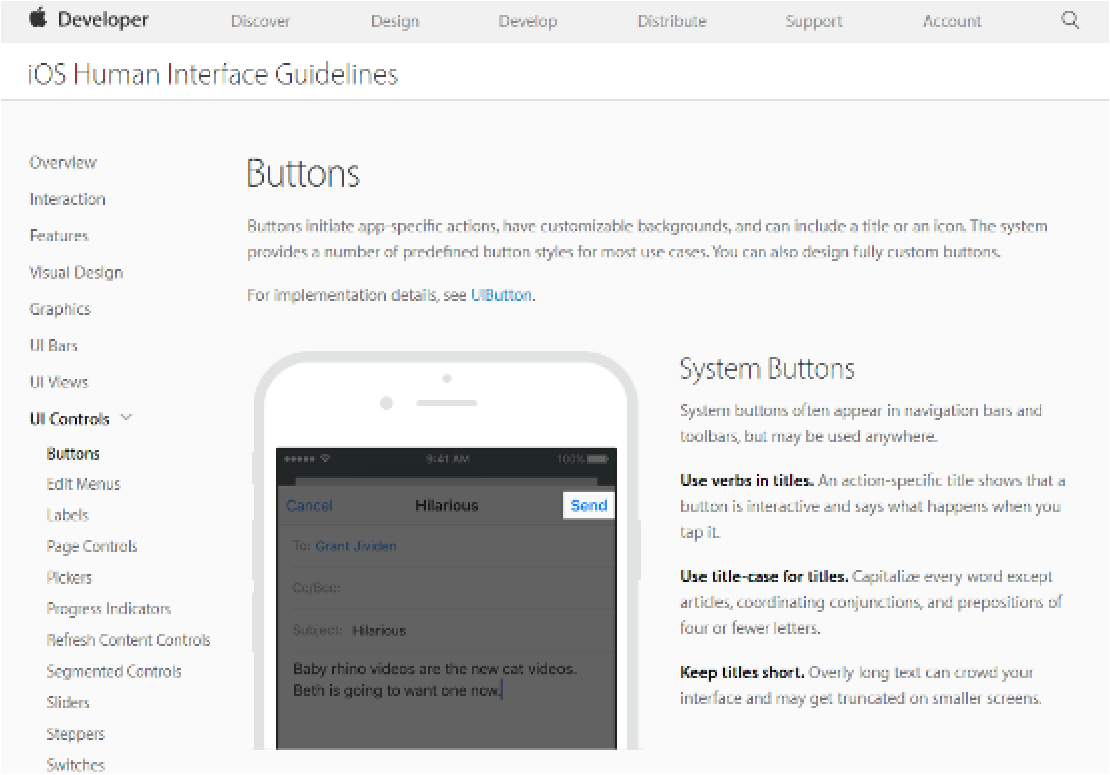


Quelle: https://youtu.be/CbGaddIZzVs

DIE GRÖßE DER LINKS

- Die simple Übernahme von `<a href..>` funktioniert nicht!
- Welche Größe muss das Element haben, damit man es mit dem Finger noch erwischt? Ich habe z.B. „riesige“ Hände...
- Button vs. Text
- Beispiel: die iOS-Human-Interface-Guidelines schreiben eine Mindestgröße von 44x44 Pixel für Touchziele vor.
  - vgl. https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/LayoutandAppearance.html

IOS HUMAN INTERFACE GUIDELINES



Quelle: https://developer.apple.com/ios/human-interface-guidelines/ui-controls/buttons/

KEIN HOVER

- Ohne Maus gibt es i.d.R. auch kein Hover. Das heisst, ein anderes Navigationskonzept muss her.
- Doppeltap ist irgendwie unpraktisch (aber nicht selten)
- Automatik mittels Anker-Ziel scheint schon besser...
- Menüstrategie überdenken!

DIE REICHWEITE

- Die Maus erreicht mit geringem körperlichen Aufwand alle Bereiche auf dem Desktop.
- Nehmen Sie jetzt einmal Ihr Handy in die Hand.
  - Steuern Sie mit den Daumen? Mit dem Zeigefinger?
  - Welche Bereiche erreichen Sie leicht und wo müssen Sie ggf. das Gerät anders halten? Machen Sie dieselbe Übung mit dem Tablet.
- Es gibt auf Smartphone und Tablet unterschiedlich schwer erreichbare Bereiche. Platzieren Sie Elemente der Interaktion mit Sorgfalt.

IMPLEMENTIERUNG

- Tablets unterscheiden sich in der Auflösung kaum mehr von Laptops oder Desktops.
- Ist das jetzt Touch oder Maus?
- Frameworks wie *Modernizr* verwenden Javascript, um die Bedienung abzufragen.

GUIDELINES

Lesen Sie doch einmal in die Guidelines von Apple, Microsoft und Co. hinein. Sie werden hilfreiche Designrichtlinien finden.

- iOS Human Interface Guidelines
- BBC: 2013 a story of how we built responsive bbc news
- https://msdn.microsoft.com/en-us/library/windows/apps/hh465424.aspx

ZUGÄNGLICHKEIT

BARRIEREFREIHEIT

BARRIEREFREIHEIT

- Zugänglichkeit ist für viele Menschen eine Frage der Barrierefreiheit. Das Internet darf betroffenen Personen nicht vorenthalten bleiben.
- Siehe auch: *Verordnung zur Schaffung barrierefreier Informationstechnik nach dem Behindertengleichstellungsgesetz (Barrierefreie-Informationstechnik-Verordnung - BITV 2.0)*
- Denken Sie z.B. auch an Behördendienste im Internet!

AUSZUG AUS DER BITV

§ 2 Einzubeziehende Gruppen behinderter Menschen

Die Gestaltung der in § 1 genannten Angebote der Informationstechnik ist dazu bestimmt, behinderten Menschen im Sinne des § 3 des Behindertengleichstellungsgesetzes, denen ohne die Erfüllung zusätzlicher Bedingungen die Nutzung der Informationstechnik nur eingeschränkt möglich ist, den Zugang dazu zu eröffnen.
Nichtamtliches Inhaltsverzeichnis

§ 3 Anzuwendende Standards

(1) Die in § 1 genannten Angebote der Informationstechnik sind nach der Anlage 1 so zu gestalten, dass alle Angebote die unter Priorität I aufgeführten Anforderungen und Bedingungen erfüllen. Weiterhin sollen zentrale Navigations- und Einstiegsangebote zusätzlich die unter Priorität II aufgeführten Anforderungen und Bedingungen berücksichtigen.
(2) Auf der Startseite des Internet- oder Intranetangebotes (§ 1 Nummer 1 und 2) einer Behörde im Sinne des § 1 Absatz 2 Satz 1 des Behindertengleichstellungsgesetzes sind gemäß Anlage 2 folgende Erläuterungen in Deutscher Gebärdensprache und in Leichter Sprache bereitzustellen:

1. Informationen zum Inhalt,
2. Hinweise zur Navigation sowie
3. Hinweise auf weitere in diesem Auftritt vorhandene Informationen in Deutscher Gebärdensprache oder in Leichter Sprache.

Die Anforderungen und Bedingungen der Anlage 1 bleiben unberührt.
Nichtamtliches Inhaltsverzeichnis

§ 4 Umsetzungsfristen für die Standards

(1) Die in § 1 genannten Angebote, die bis zum 22. März 2012 neu gestaltet oder in wesentlichen Bestandteilen oder größerem Umfang verändert oder angepasst werden, sind nach § 3 zu erstellen. Mindestens ein Zugangspfad zu den genannten Angeboten soll mit der Freischaltung dieser Angebote die Anforderungen und Bedingungen der Priorität I der Anlage 1 erfüllen.
(2) Angebote nach § 1 Nummer 1 und 2, die vor dem in Absatz 1 Satz 1 genannten Stichtag veröffentlicht wurden, sind spätestens bis zum 22. September 2012 nach § 3 Absatz 1 zu gestalten. Sie sind zusätzlich spätestens bis zum 22. März 2014 nach § 3 Absatz 2 zu gestalten.
(3) Für Angebote nach Absatz 2 gilt bis zur Umsetzung im Sinne der Absätze 1 und 2 die Barrierefreie-Informationstechnik-Verordnung vom 17. Juli 2002 (BGBl. I S. 2654) fort.
Nichtamtliches Inhaltsverzeichnis

Quelle: http://www.gesetze-im-internet.de/bitv_2_0/BJNR184300011.html [am 20.10.2016]

WAHRNEHMBARKEIT

- Eine Webseite muss gut sehbar, hörbar und fühlbar sein.
- Schriftarten und -größen muss der Nutzer einstellen können.
- Farben und Kontraste müssen verwendbar sein.
  - rot-grün Schwäche, Grauer Star
- Screenreader, Braille-Anzeigegeräte,...

BEDIENBARKEIT

- Interaktion muss allen Menschen möglich sein (schlechtes Sehvermögen, motorische Einschränkungen).
- Navigation z.B. alternativ per Tastatur realisiert
- Video-/Audiosteuerungselemente
- Links gut sichtbar und ausreichend groß auf Handys

VERSTÄNDLICHKEIT

- Schreiben Sie leicht verständlich! Keine komlizierten Sätze und Inhalte.
- Intuitive Verständlichkeit ist subjektiv.
- Verwenden Sie Icons mit Bedacht und führen Sie keine eigenen Standards ein.
- Lernen Sie von Best-Practise und seien Sie offen für Veränderungen.

ROBUSTHEIT

- Informationstechnologie soll auch dann funktionieren, wenn eine oder mehrere Teile nicht vollständig oder wie geplant arbeiten.
- Bei der Entwicklung von Webseiten wird das insbesondere auf die Unterstützung von Technologien zutreffen.
  - Javascript, Media Queries, HTML-Standard, etc.
- Prüfen Sie also nach Möglichkeit den Client auf Evantualitäten und bieten Sie dem Anwender eine Fall-Back-Lösung an.

MEIN PERSÖNLICHER TIPP

Gute Entwickler implementieren nicht das was ihnen besonders gefällt, sondern das was Nutzer brauchen.

Das kann auch bedeuten, dass die Nutzer noch nichts davon wissen ☺ (vgl. Steve Jobs, Henry Ford).

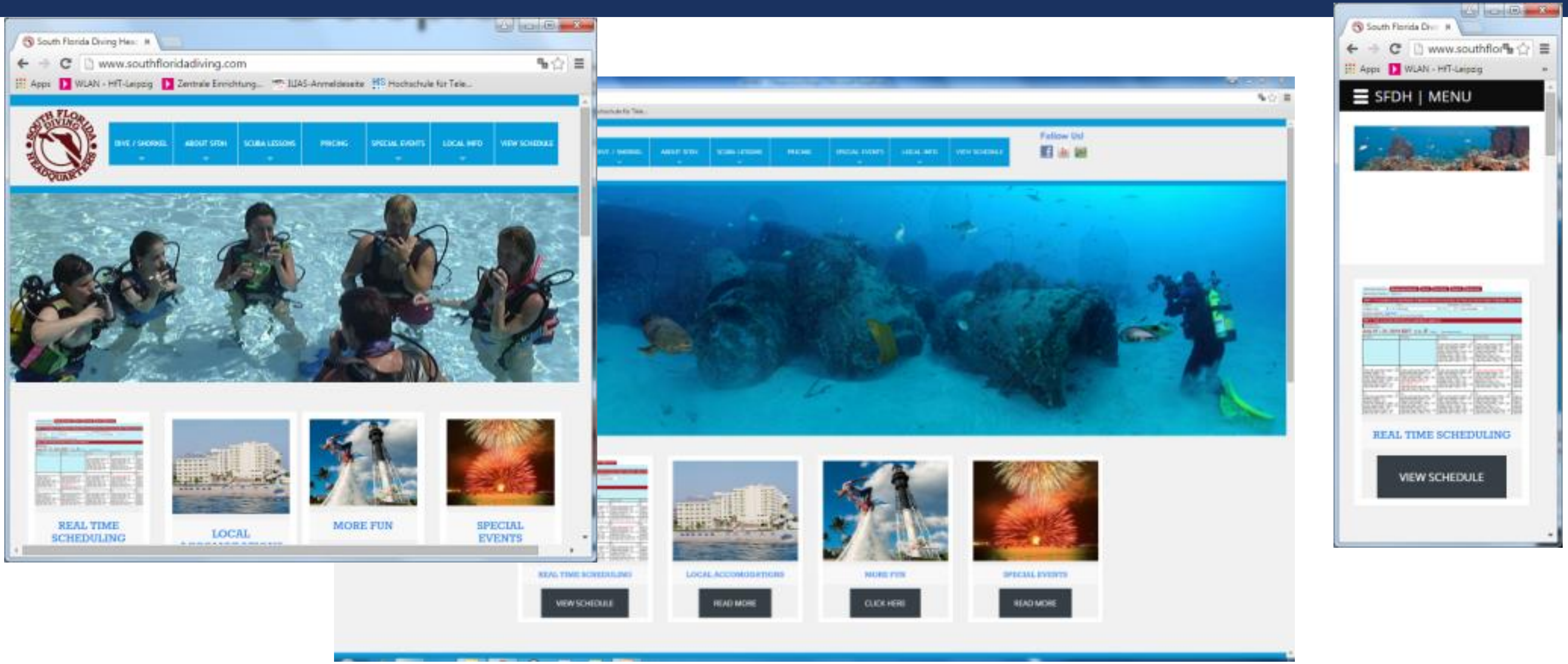
Vgl. auch: User Experience – ISO 9241-210

BREAKPOINTS

RESPONSIVE DESIGN

NOCHMALS…

http://www.southfloridadiving.com/



LAYOUT UMBRECHEN

Die Definition ab wann der Browser aufgrund einer Media Query ein alternatives, vorliegendes Layout rendert, bezeichnen wir als Breakpoint.

AUSWAHL

- In den meisten Fällen werden Sie Breakpoints auf Basis der Viewports definieren.
- Es gibt keine allgemein gültige Regel.
- Erfolgreiche Entwickler…
  - Konzentrieren sich zunächst auf den Inhalt
  - Beginnen mit dem größten oder dem kleinsten Layout
  - Vergrößern/Verkleinern soweit, bis sie nicht mehr zufrieden sind und
  - Setzen ebendort einen Breakpoint

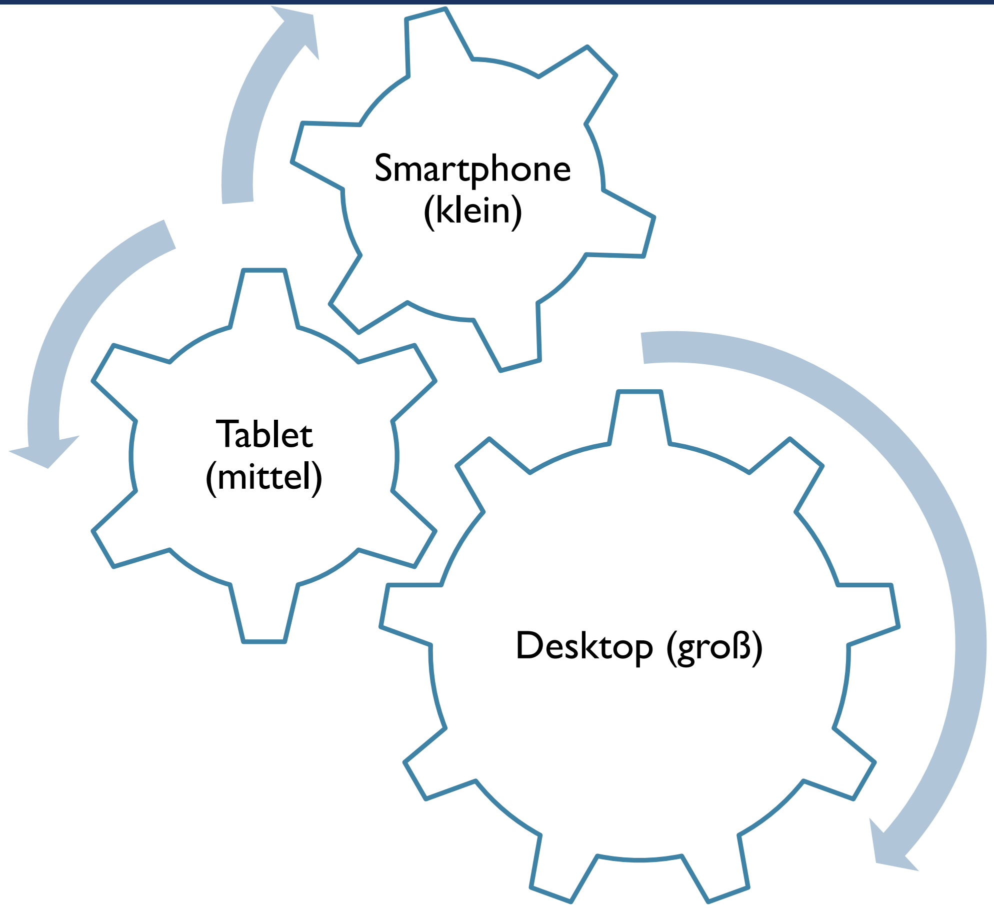
ANZAHL

- Komplexer Inhalt = komplexe Webseite = viele Breakpoints?
- Viele Breakpoints sind teuer in der Entwicklung!
- Zu wenig Breakpoints können Nutzer abschrecken!

Mein Tipp:

„So wenig wie möglich, so viel wie notwendig.“

IN DER ÜBUNGSAUFGABE



LAYOUT-PATTERNS

RESPONSIVE DESIGN

LAYOUT-PATTERNS

Übersicht einiger bekannter Muster:

Tiny Tweaks (marginale Anpassungen)	Mostly Fluid (meistenteils fließend)	Column Drop (Spaltenumbruch)	Layout-Shifter (Layoutverdreher)	Off-Canvas-Layout (außerhalb des sichtbaren Bildschirminhalts)	Footer-Navigation und Off-Canvas-Marginalie	Top-Off-Canvas-Menü und Off-Canvas-Marginalie	Vertikale und horizontale Off-Canvas-Panels

TINY TWEAKS

- einspaltiges Layout auf allen Medien
- minimale Anpassungen: Schriftgröße (font-size), Ränder (padding)

MOSTLY FLUID

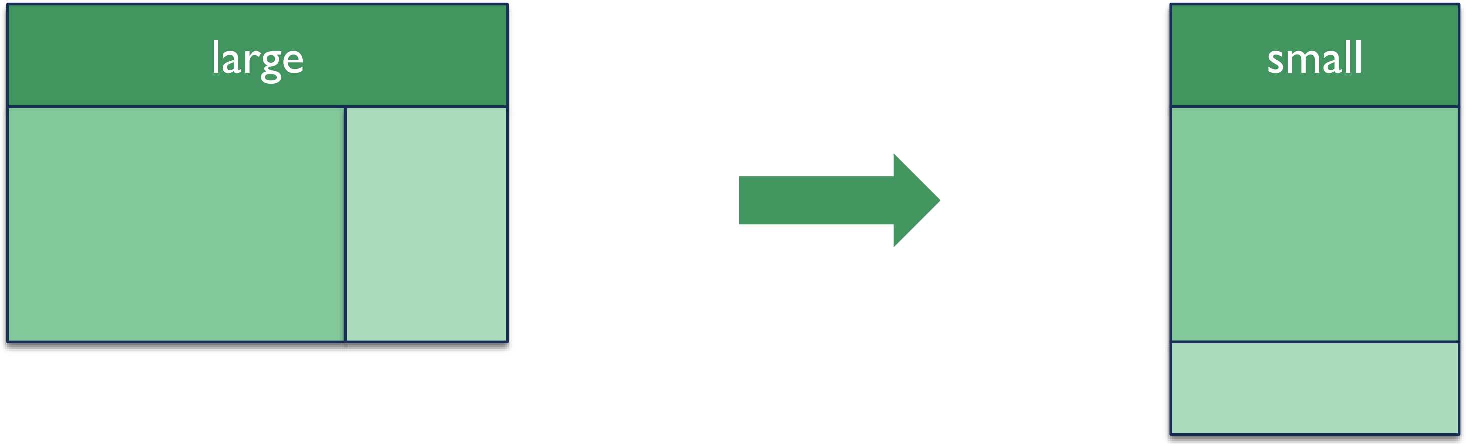
- mehrspaltige Layouts
- variabel Anpassung von `margin`
- Skalierung der Inhalte (Bilder)

COLUMN DROP

- mehrspaltiges Layout
- im Unterschied zu Mostly Fluid bleibt die Größe der Inhalte nach Möglichkeit erhalten
- wird der Viewport kleiner, so „rutschen“ Spalten in die nächste Zeile (bis nur noch eine Spalte da ist)
- Breakpoints müssen mit dem Layout abgestimmt werden

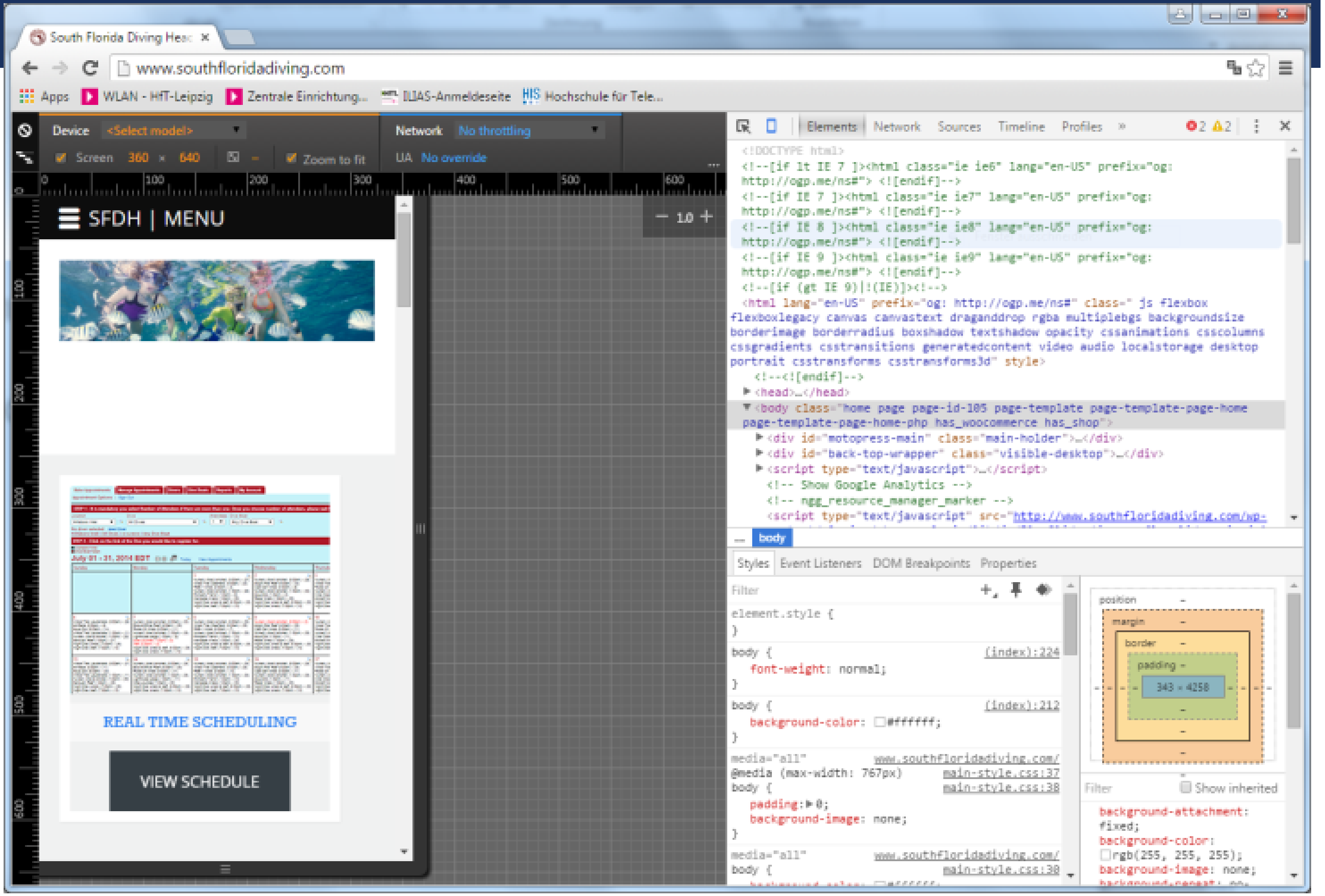
LAYOUT SHIFTER

- mehrspaltige Layouts
- für mehrere Viewports (z.B. small, medium, large) wird das Layout in der Anordnung umgestaltet



WERKZEUGE

- Unterstützung in vielen Browser (Emulatoren)
    - Firefox, Chrome
- Tools zur automatischen Erstellung von Media Queries



CSS UNTERSTÜTZUNG

LAYOUT-TECHNIKEN

FLEXBOX-LAYOUT

- Siehe Folien zu CSS.
- Steuerung der Breite + Reihenfolge mittels `flex` und `order`
- Wechsel der Ausrichtung mittels `flex-direction`
- Einsatz z.B. in Media Queries (vgl. Layout-Shifter)

```
.containerClass {
   display: flex;
   flex-direction: column; /*row zum wechseln*/
}

.elementClass {
   flex: 3; /* entspricht ca. 30% der Bildschirmbreite*/
   order: 1;
}
```

FRAMEWORKS FÜR RESPONSIVE DESIGNS

HIER BEGINNT DIE REALE WELT…

Ein Freund von mir sagte es so: „…*heutzutage entwickelt doch keine Firma mehr manuell Webseiten oder Javascript. Das passiert alles nur noch auf Frameworks!*“

Leider entwickeln sich die Frameworks mit rasanter Geschwindigkeit.

In der Folge entstehen unzählige Abhängigkeiten und Versions-Inkompatibilitäten!

UNTERSCHIEDLICHE AUSPRÄGUNG

Minimalistische Frameworks

- Als Unterstützung gezielter Funktionen und Layoutkomponenten

Umfangreiche Frameworks

- Mit vordefinierten Stylesheets und Klassen

Hilfreiche Fragen zur Auswahl:

- [ ] Einsatzplanung: spezielles Projekt oder tägliche Arbeit?
- [ ] Kontext: große/kleine Webseite, App, …?
- [ ] Was bietet das Framework an Funktionalitäten?
- [ ] Spielen Barrierefreiheit und Semantik eine Rolle?
- [ ] Was ist mit CSS-Präprozessoren, wie LESS, SASS?
- [ ] Sind Modularität und Downloadgröße wichtiger?
- [ ] Individuelles oder Standard-Layout?

EIN KLEINES BEISPIEL: YAML

- *Yet Another Multicolumn Layout*
- Kommt mit Standards für:
  - feste, flexible und rasterbasierte Layouts
  - Tabellen, Formulare, Navigationsmenüs
  - „YAML-Builder“
- Einbindung mittels:
  - vorgefertigte CSS-Dateien
  - Verwendung entsprechender Klassennamen im HTML-Dokument

Responsive Grids

Flexible layouts work best when their responsive behavior is based on the boundary conditions of the individual screen designs. That's why YAML doesn't provide a pre-baked plan for linearization of grid elements.

As a frontend developer, you are completely free to define, how many levels of linearization your design needs and when they should be applied.

Progressive Linearization

Here's a code-snippet to a linearization level for the grid module at a viewport size, lower then 760px. To create additional levels, copy this media block, set the viewport size, when the new level is triggerd (*max-width*) and count up the number in the class name.

text pop-up

```
@media screen and ( max-width: 760px ) {

  /* linearization for grid module */
  .linearize-level-1,
  .linearize-level-1 > [class*="ym-g"] {
    display: block;
    float: none;
    padding: 0;
    margin: 0;
    width: 100% !important;
  }

  /* reset defined gutter values */
  .linearize-level-1 > [class*="ym-g"] > [class*="ym-gbox"] {
    overflow: hidden; /* optional for containing floats */
    padding: 0;
    margin: 0;
  }
}
```

Add the class `.linearize-level-1` to any grid module, you want to be linearized:

```
<div class="ym-grid linearize-level-1">
  ...
</div>

```

YAML - EXAMPLE

Quelle: http://www.yaml.de/docs/index.html#yaml-grids (22.10.2015)

EIN WEITERES BEISPIEL: PURECSS

- Minimalistischer Ansatz – Stylesheets liegen bei 4 Kbyte
- Yahoo, erweiterbar mit *Pure Extras*
- Standards für
  - Responsive Layouts
  - Formulare, Buttons, Menüs
- Die Einbindung erfolgt analog mittels CSS und Klassen.

Next, let's look at a responsive grid. Elements within this grid will be `width: 100%` on small screens, but will shrink to become `width: 33.33%` on medium-sized screens and above.

```
<div class="pure-g">
    <div class="pure-u-1 pure-u-md-1-3"> ... </div>
    <div class="pure-u-1 pure-u-md-1-3"> ... </div>
    <div class="pure-u-1 pure-u-md-1-3"> ... </div>
</div>
```

Default Media Queries

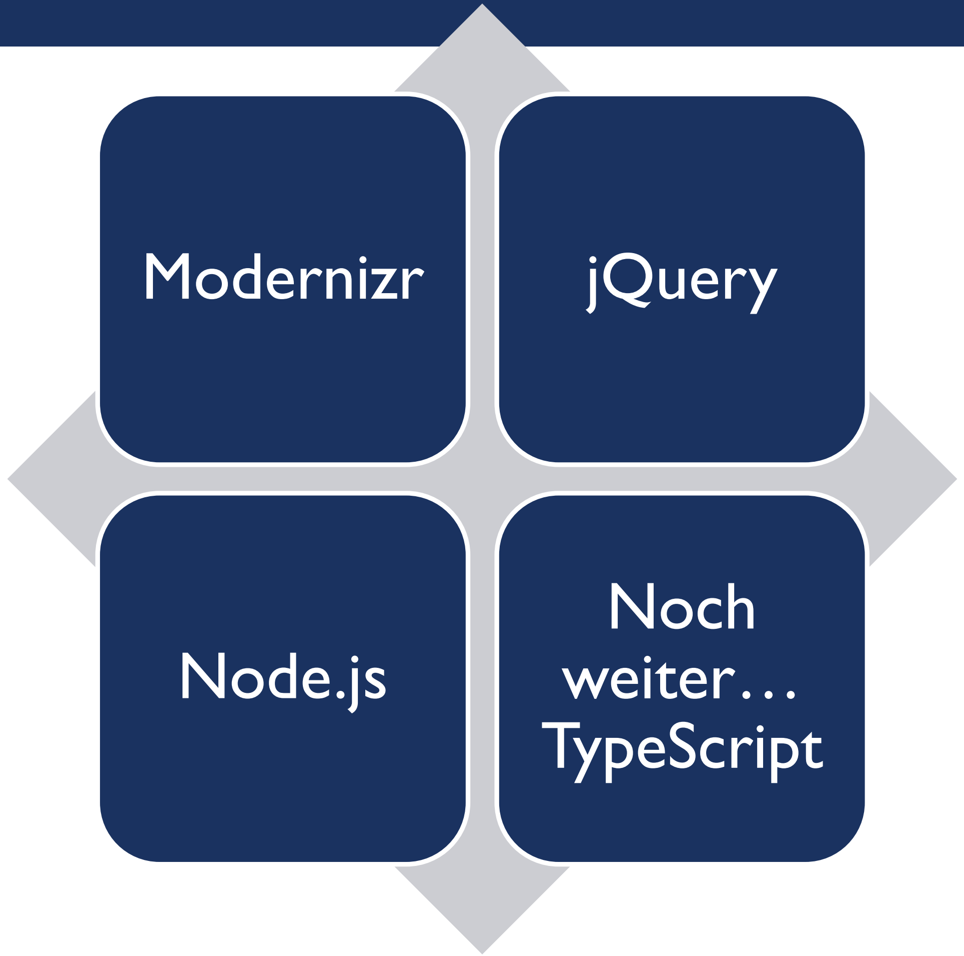
When using Responsive Grids, you can control how the grid behaves at specific breakpoints by adding class names. Pure's default responsive grids comes with the following class names and media query breakpoints.

Key	CSS Media Query	Applies	Classname
None	*None*	*Always*	`.pure-u-*`
`sm`	`@media screen and (min-width: 35.5em)`	≥ 568px	`.pure-u-sm-*`
`md`	`@media screen and (min-width: 48em)`	≥ 768px	`.pure-u-md-*`
`lg`	`@media screen and (min-width: 64em)`	≥ 1024px	`.pure-u-lg-*`

PURECSS - EXAMPLE

Quelle: http://purecss.io/grids/ (22.10.2015)

…AUCH FÜR SCRIPT (ABER DAS ÄNDERT SICH SEHR SCHNELL)



NAVIGATIONSKONZEPTE

RESPONSIVE DESIGN

NUTZERFREUNDLICHE NAVIGATION

Schlechte Beispiele

- Komplexität
- Inkonsistenz
- „Design follows IT“
- Unverständlich
- „Death-Lock‘s“
- Tote Links (404)
- Schlechte Semantik!!!

Gute Beispiele

- Listenform
- Erkennbar & Zugänglich
- Intuitiv
- Konsistenz
- „Menschlich“
- Vollständigkeit
- Gute Semantik!!!

TOUCH ME

Besonders wichtig bei mobilen Geräten – es gibt keine Maus und damit auch kein Hover!

Buttondesign anpassen für Touchsteuerung (Größe und Optik)

Alternativen für Ausklappmenüs implementieren

z.B. 1st touch für Ausklappen, 2nd touch für Aktivierung

PATTERNS

- Fließendes „Menüband“ oben
  - Kann jedoch Inhalt verdecken, wenn zuviele Einträge.
- Ein Menübutton verlinkt an das Ende der Seite.
- Menübutton öffnet Auswahlbox.
  - Keine Lösung für verschachtelte Menüs.
- Off-Canvas Menü (Slider links oder oben)
- Toggle-Menü klappt komplette Liste auf.

ANPASSUNGSFÄHIGE ELEMENTE

RESPONSIVE DESIGN

BILDER

- Skalierung haben wir schon kennengelernt.
- Manchmal ist es besser, nur einen Ausschnitt anzuzeigen anstelle zu skalieren.
  - Einen Container einsetzen (`<figure>`), um das Bild darin mit 100% Größe auf den Ausschnitt zu verschieben.
- Mehrere Bildkopien in unterschiedlichen Auflösungen.
  - Siehe auch den Einsatz von `<picture>` und das Attribut `srcset` für `<img>`

HINTERGRUND

- Einzelne Bilder für jedes Layout
- Kacheln
- Icons
- Skalierung (Proportion beachten: cover vs. contain)

SVG

- Bitmap-Grafiken haben ein festes Seitenverhältnis und eine feste Größe.
- Besser skalieren lässt es sich mit Vektor-Grafiken.
- SVG wurde eben dafür erfunden.
- Illustrationen – ja / Photos – nein

VIDEOS (HTML5)

HTML

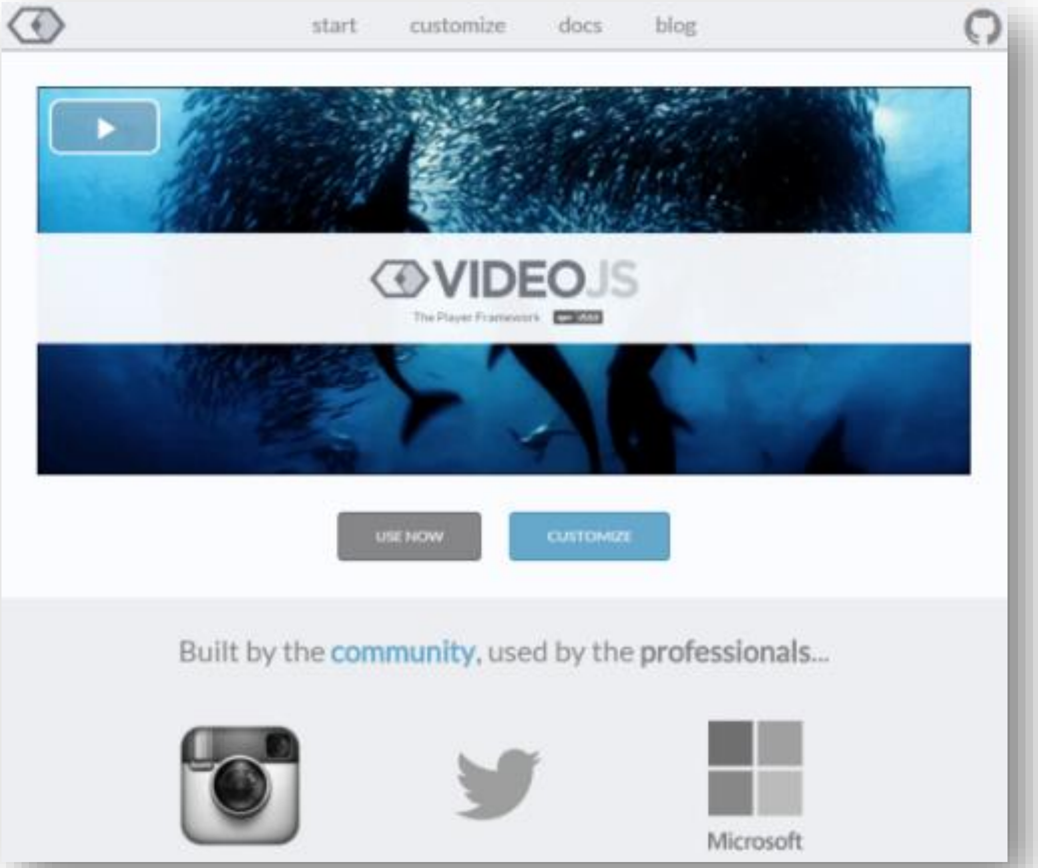
```
<video poster=„vorschau.png“ controls tabindex=„0“>
 <source src=„video.mp4“ type=„video/mp4“></source>
 <p> Ihr Browser unterstützt keine Videos </p>
</video>
```

CSS

```
video {
 max-width: 100%;
 heigth: auto;
}
```

Kein Problem mehr mit HTML5

Analog zu Bildern



SCRIPT-ALTERNATIVE MIT VIDEO.JS

Quelle: http://videojs.com/ (22.10.2015)

ENDE