

# ***RESPONSIVE WEBDESIGN WORKSHOP 2017***

BY DOMINIC VOGL



# KURZ ZU MEINER PERSON

Hallo,

mein Name ist **Dominic Vogl**, ich bin 27 Jahre alt und arbeite für sigikid in Mistelbach. Dort bin ich als Webdesigner und Front-End Entwickler für E-Commerce und die digitale Markenführung mitverantwortlich. Mein Hauptaugenmerk liegt dabei in der technischen und optischen Weiterentwicklung unseres Webshops, sowie der Corporate Website. Zur digitalen Markenführung gehören unsere Social Media Auftritte, Newsletter oder auch die Fotografie von Imagebildern.

Kurzvita:

2008 - C3 marketing Agentur, Ausbildung zum Mediengestalter Digital & Print

seit 2011 - selbstständiger Webdesigner, Front-End Entwickler, und Trainer

2012 - 4c media, Webdesigner / Front-End Entwickler

2015 - BERGWERK Werbeagentur, Webdesigner / Teamleader Online

2015 - sigikid, Webdesigner / Front-End Entwickler

2017 - sigikid, Teamleader E-Commerce / Ausbilder

# ***DAS ERGEBNIS DER SCHULUNG***

Ziel des Workshops ist es, dass jeder Teilnehmer

- ▶ ... mit grundlegenden HTML Tags Inhalte struktuiert
- ▶ ... mit einfachen CSS Anweisungen die Darstellung der Inhalte manipuliert
- ▶ ... eine einfache HTML Website baut und diese dann...
- ▶ ... in responsiv umsetzt
- ▶ ... die Merkmale des Responsive Webdesigns kennen lernt
- ▶ ... sein Template mit verschiedenen Elementen oder Scripten um neue Funktionen erweitert



***Kursergebnisse / Dokumente / Fragen & Antworten***

***<https://fb.com/groups/777701422406686/>***

# AGENDA

## ► **Vorwort**

- Kurze Vorstellung
- Ergebnispräsentation
- Timeline

## ► **Einführung HTML5**

- Syntax
- Webstandards und Grundlagen
- Inhalte auszeichnen und strukturieren
- Das Document Object Model (DOM)
- Webdeveloper Tools
- Validatoren und Hilfsmittel
- Bilder und Grafiken
- Semantik, Barrierefreiheit und Suchmaschinenoptimierung

## ► **Einführung in CSS3**

- Syntax und Semantik
- HTML und CSS verbinden
- Vererbung
- Die Kaskade
- Das CSS Box Model
- Element Positionierung

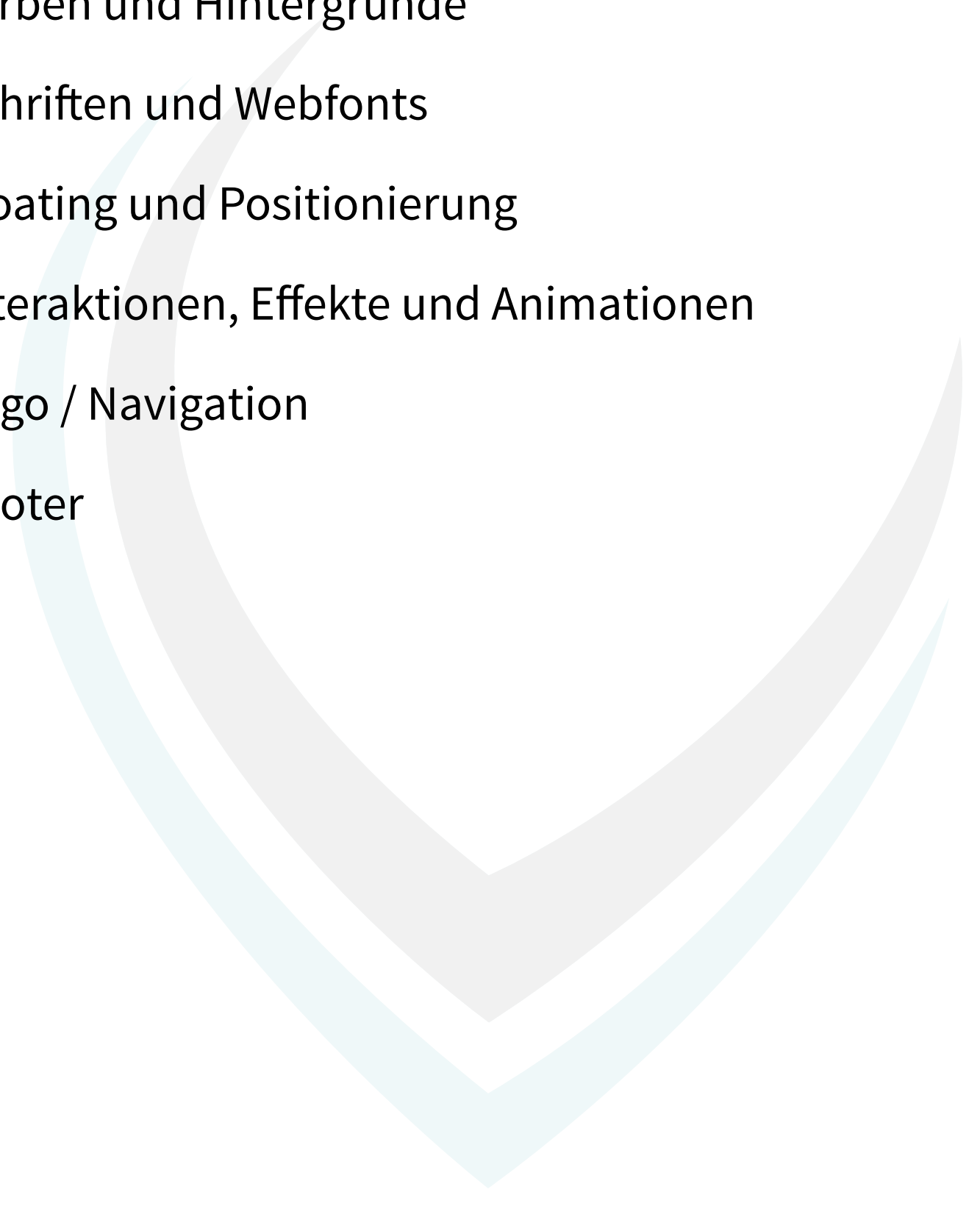
# AGENDA

## BEISPIELPROJEKT: SCHRITT FÜR SCHRITT

### ► Part I: HTML

1. Das HTML-Grundgerüst entwickeln
2. Inhalte strukturieren
  - a) Texte und Bilder
  - b) Listen und Navigationselemente
3. Verlinkungen und Anker

### ► Part II: CSS

4. Farben und Hintergründe
  5. Schriften und Webfonts
  6. Floating und Positionierung
  7. Interaktionen, Effekte und Animationen
  8. Logo / Navigation
  9. Footer
- 

# AGENDA

## RESPONSIVE WEBDESIGN

- 10. Responsive Web Design Grundlagen
- 11. Klassischer & skalierter Viewport
- 12. Breakpoints & Media Queries
- 13. Adaptive Layout & Responsive Layout
- 14. »Mobile First« & »Content First«
- 15. Flexible Gestaltungsraster (Grids) entwickeln
- 16. Ein verändertes Box-Modell
- 17. Frameworks: Bootstrap, Foundation & Co.

- 18. Grafiken im Responsive Design  
– picture, srcset & Co.
- 19. Videos im Responsive Design
- 20. Tabellen im Responsive Design
- 21. Javascript
- 22. Slider
- 23. Image-Feed



# ARBEITSDATEN

Github kann man als Social Media Plattform für Entwickler ansehen. Man kann hier seinen Code versionieren, sichern, mit anderen teilen, im Team zusammenarbeiten oder auch Code von anderen nehmen und selbst weiterentwickeln.

Alle Kursdateien finden sich ebenfalls auf Github und jeder das einen Account hat kann sich davon einen Fork erstellen und selbst am Code weiterarbeiten. Dieses Projekt wird von mir auch in regelmäßigen Abständen aktualisiert, korrigiert und erweitert.



***Arbeitsdaten auf Github.com***

***[https://github.com/dominicvogl/workshop\\_responsive\\_webdesign](https://github.com/dominicvogl/workshop_responsive_webdesign)***

## QUELLEN:

► <https://github.com>

# ***EINFÜHRUNG IN HTML***





# WAS IST HTML?

Die **Hypertext Markup Language** (HTML, dt. Hypertext-Auszeichnungssprache), ist eine textbasierte Auszeichnungssprache zur Strukturierung von Inhalten wie Texten, Bildern und Hyperlinks in Dokumenten. HTML-Dokumente sind die Grundlage des World Wide Web und werden von einem Webbrowser dargestellt. Neben den vom Browser angezeigten Inhalten einer Webseite enthält HTML zusätzliche Angaben in Form von Metainformationen, die z. B. über die im Text verwendete Sprache oder den Autor Auskunft geben oder den Inhalt des Textes zusammenfassen.



## Beispiel

***01\_HTML/01\_syntax.html***

## QUELLEN:

- ▶ [https://developer.mozilla.org/de/Learn/HTML/Introduction\\_to\\_HTML/Getting\\_started](https://developer.mozilla.org/de/Learn/HTML/Introduction_to_HTML/Getting_started)
- ▶ <http://www.webdesign-cheatsheets.com/html5.html>
- ▶ <https://websitesetup.org/wp-content/uploads/2014/02/HTML-CHEAT-SHEET.png>

# WEBSTANDARDS UND GRUNDLAGEN

Ein jedes valides HTML Dokument besteht aus gewissen Elementen, welche zum Grundgerüst gehören, da diese essenziell für den Browser sind um sichtbare und unsichtbare Inhalte voneinander zu trennen und um klar zu deklarieren, ab wann bestimmte Inhalte beginnen, wo Dateien eingebunden oder Meta-Tags definiert werden.



**Beispiel**

***01\_HTML/02\_grundgeruest.html***

## QUELLEN:

- ▶ <https://www.w3.org/>
- ▶ <http://www.webdesign-cheatsheets.com/html5.html>

# INHALTE AUSZEICHNEN



**Beispiel**

***01\_HTML/03\_struktur.html***

## QUELLEN:

► <http://www.webdesign-cheatsheets.com/html5.html>



# ***DAS DOCUMENT OBJECT MODEL (DOM)***

**Document Object Model (DOM)** - Eine Implementierung, die dieser Spezifikation genügt, besteht im Sinne der objektorientierten Programmierung aus einem Satz von Klassen zusammen mit deren Methoden und Attributen. Sie erlaubt Computerprogrammen, dynamisch den Inhalt, die Struktur und das Layout eines Dokuments zu verändern.



***Beispiel***

***Direkt im Browser***

## **QUELLEN:**

► [https://de.wikipedia.org/wiki/Document\\_Object\\_Model](https://de.wikipedia.org/wiki/Document_Object_Model)

# WEBDEVELOPER TOOLS / VALIDATOREN

Jeder gängige Browser, wie z.B. Google Chrome oder Mozilla Firefox liefern standardmäßig diverse Entwickler-tools, mit welchen sich der Code von jeder Website untersuchen und nachvollziehen lassen kann.



***Beispiel***

***Direkt im Browser (Chrome, Firefox)***



***Beispiel***

***Validatoren (W3C)***

## QUELLEN:

- ▶ <https://www.google.de/chrome/>
- ▶ <https://www.mozilla.org/de/firefox/>
- ▶ <https://validator.w3.org/>

# BILDER UND GRAFIKEN

Das World Wide Web lebt natürlich nicht nur von seinen Textinhalten, sondern vor allem auch von den verschiedensten Bildern, welche aus Fotos, Grafiken oder auch Strichzeichnungen bestehen können. Für all diese Bildarten gibt es verschiedene Bild & Dateiformate, welche man im Web einsetzen kann. **.jpg, .gif, .svg, .png** um die Gängigsten zu nennen, so hat jedes seinen speziellen Eigenschaften, welche für verschiedene Einsatzgebiete eine optimale Lösung bietet.



**Beispiel**

***01\_HTML/04\_images-graphics.html***

# ***SEMANTIK, BARRIEREFREIHEIT UND SUCHMASCHINENOPTIMIERUNG***

Semantik, auch Bedeutungslehre, nennt man die Theorie oder Wissenschaft von der Bedeutung der Zeichen. Zeichen können in diesem Fall Wörter, Phrasen oder Symbole sein.

In unserem Fall sind das die HTML Tags und der darin enthaltene Text. Bei unserer HTML Struktur geht es dabei vor allem um die Bedeutung der Tags und das Zusammenspiel mit anderen Code Abschnitten um sinnvollen, validen und logischen Code zu erzeugen.



***Beispiel***

***01\_HTML/05\_semantik-seo.html***

# ***BEISPIELPROJEKT***

## ***PART I: HTML***





# HTML GRUNDGERÜST

Das Projekt wird wiederum mit einem validen HTML Grundgerüst gestartet, so wie es bei jedem modernen Webprojekt der Fall sein sollte.



**Beispiel**

***03\_RWD/01\_grundgeruest/***

# INHALTE STRUKTURIEREN

Zu Anfang definieren wir erste Strukturelle Elemente z.B. für einen Blogpost. Dieser sollte eine Überschrift, ein Datum, einen Anreissertext, „Weiterlesen“ Button und ein paar Listenelemente z. B. für Schlagworte beinhalten. Ausserdem beinhaltet der Blogpost noch ein Beitragsbild.

Zudem benötigen wir auch eine Struktur für die verschiedenen Navigationspunkte um durch die Website zu navigieren, hierbei können wir wiederum eine Liste verwenden.



## **Beispiel**

***03\_RWD/02a\_structure/  
03\_RWD/02b\_navigation/***

# VERLINKUNGEN UND ANKER

Neben Verlinkung von einzelnen Unterseiten in Form einer Navigation, kann man unter anderem auch auf andere Websites verlinken. Klassisch bei einem Blog wären Profile auf Social Media Plattformen oder Business Netzwerken. Den Zielen, zu welchen ein Anker hinführt sind wenig Grenzen gesetzt. Bei jedem Link kann man dann noch über Attribute entscheiden ob die Seite in einem neuen Fenster / Tab geöffnet werden soll oder nicht.



**Beispiel**

***03\_RWD/03\_anker/***

# ***EINFÜHRUNG IN CSS***



# SYNTAX UND SEMANTIK

„**Cascading Style Sheets**“ dienen der optischen Modifizierung verschiedenster Elemente eines HTML Dokuments. Dabei können alle möglichen Attribute wie Farbe, Schriftgröße, Hintergrundbild, Abstände, Ränder und vieles mehr gestaltet werden. So können HTML Tags direkt, über ID's oder Klassen angesprochen werden.

Man kann CSS wie Absatzformate in InDesign verstehen. Man gibt verschiedenen Elementen jeweils Formatierungen um eine einheitliche Optik zu erreichen und eine schnelle Anpassung zu ermöglichen.



**Beispiel**

***02\_CSS/01\_syntax.html***

# SYNTAX UND SEMANTIK

Code ist qualitativ hochwertig, wenn er valide, logisch aufgebaut und einfach zu lesen ist.

## Validität

- ▶ Code ist valide wenn er dem W3C Standard entspricht (<http://validator.w3.org/check>)
- ▶ Valider Code vermeidet Fehldarstellungen in älteren Browsern und ist gut für das Google Ranking

## Logik

- ▶ Beachtung der Semantik
- ▶ Benennung von Klassen und ID nach Funktion und nicht nach aussehen

## Lesbarkeit

- ▶ Saubere Einrückungen



**Beispiel**

***02\_CSS/01\_syntax.html***

# CSS SELEKTOREN

Selektor: a, div, p, span	Hier wird direkt ein HTML Tag angesprochen. Dabei sind alle HTML Tags die es gibt möglich.
#ID's: #header, div#navigation	Hier wird nur das Element mit der jeweiligen #ID angesprochen
.Klassen: .zitat, span.highlight	Hier werden alle Elemente mit der jeweiligen .Klasse angesprochen
:Pseudoklassen .zitat:hover span.highlight:active	Hier wird das Element mit der jeweiligen Klasse und über dem sich der Mousezeiger befindet, bzw angeklickt wurde angesprochen
/* ... */	Kommentare in CSS

## QUELLEN:

- <http://code.tutsplus.com/tutorials/the-30-css-selectors-you-must-memorize--net-16048>

# HTML UND CSS VERBINDEN

externe Einbindung

```
<link href="css/extern.css" rel="stylesheet" type="text/css"
media="screen" />
```

interne Einbindung

```
<style type="text/css">
    div#container1 { background: red; }
</style>
```

inline Einbindung

```
<div style="width: 400px; color: white;">Container</div>
```

@import

```
@import url („import.css“);
```



**Beispiel**

***02\_CSS/02\_integration.html***



# VERERBUNG UND KASKADE

Eine besondere Eigenschaft von CSS ist die Kaskade. Kaskade bedeutet, dass ein Dokument nicht nur von einem einzelnen Stylesheet formatiert werden kann, sondern von einer Vielzahl von Stylesheets, die aus verschiedenen Quellen stammen können. Dadurch wird CSS flexibel und benutzerfreundlich, da die einzelnen Stylesheets aufeinander aufbauen. Gleichzeitig entsteht aber das Problem, dass Eigenschaften für ein Element mehrfach und mit widersprüchlichen Werten festgelegt werden können. Dieses Problem wird umgangen, indem die Kaskade für Regeln und Eigenschaften eine Gewichtung errechnet, anhand derer die tatsächlich für ein Element geltenden Formate bestimmt werden.



**Beispiel**

***02\_CSS/01\_syntax.html***

## QUELLEN:

- ▶ <https://wiki.selfhtml.org/wiki/CSS/Kaskade>
- ▶ [https://stuffandnonsense.co.uk/archives/css\\_specificity\\_wars.html](https://stuffandnonsense.co.uk/archives/css_specificity_wars.html)
- ▶ <http://www.webdesign-cheatsheets.com/css3.html>

# EINIGE GÄNGIGE CSS EIGENSCHAFTEN

```
width: 120px;  
height: 50%;
```

Höhe und Breite eines Objektes, dabei sind Angaben z.B. mit Pixel, Prozent oder EM möglich.

```
background:  
url(../elemente/bg_spacer.png)  
#e2e2e2;
```

Eigenschaften des Hintergrundes:  
Link zu einem Hintergrundbild, Hintergrundfarbe

```
display: none,  
inline,  
block,
```

ausgeblendet,  
erzwingt Anzeige im Textfluss,  
erzwingt Anzeige als Block - Umbruch

```
position:  
static,  
relative,  
absolute,  
fixed
```

normaler Textfluss,  
relative Positionierung (gemessen von Normalposition)  
absolute Positionierung  
(gemessen am Rand des Elternelements)  
fixe Positionierung (gemessen am Viewport, scrollt nicht mit)

## QUELLEN:

► <http://www.webdesign-cheatsheets.com/css3.html>

# EINIGE GÄNGIGE CSS EIGENSCHAFTEN

`font-size: 12px, 15pt, 2%, 1.2em;`

Schriftgröße, kann in Pixeln, Punkt, Prozent und EM angegeben werden

`line-height: 12px, 15pt, 2%, 1.2em;`

Zeilenhöhe, kann in Pixeln, Punkt, Prozent und EM angegeben werden

`color: #232323, rgb(255,125,100), rgba(255,210,34,0.7);`

Schriftfarbe, kann mit #HEX, RGB und RGBA angegeben werden

`font-weight: normal, light, bold`

Strichstärke der Schrift

`font-family: ,Arial``

Legt die zu verwendende Schrift fest

`z-index: 10`

bestimmt die Reihenfolge welche Elemente im Vordergrund oder Hintergrund liegen. Vergleichbar mit Photoshop Ebenen.

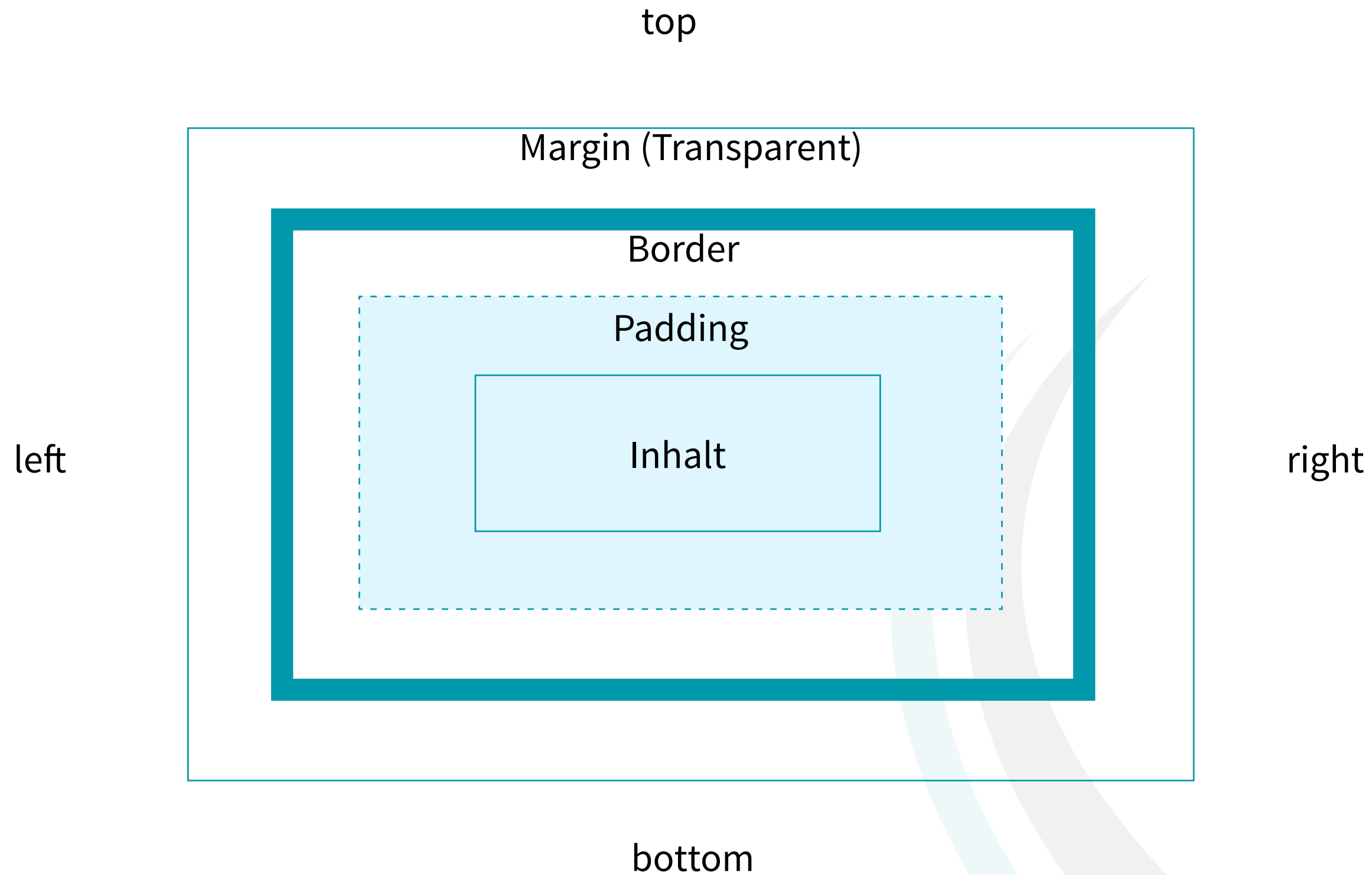
`float: left, right`

Verwandelt ein Element in ein Block-Element. Andere Elemente umfließen dieses dann.

## QUELLEN:

► <http://www.webdesign-cheatsheets.com/css3.html>

# DAS CSS BOX-MODELL



***Beispiel***

***02\_CSS/03\_box-modell.html***

# ELEMENT POSITIONIERUNG

HTML Elemente lassen sich via CSS auf verschiedene Arten und Weisen frei positionieren. Die Unterschiede zwischen den verschiedenen Positionierungen kommen bei der Wechselwirkung mit anderen Elementen zum Vorschein.

Die gängigsten Werte für die Eigenschaft „position“:

- ▶ static (default)
- ▶ relative
- ▶ absolute
- ▶ fixed



***Beispiel***

***02\_CSS/04\_positions.html***

# FARBEN UND HINTERGRÜNDE

Via CSS hat man die Möglichkeit die Farbigkeit von jeglichen Schriften und Elementen in gewisser Weise anzupassen. Dabei steht der komplette RGB Farbraum zur Verfügung. Die Werte lassen sich hier über RGB(A), Hex oder auch über Namenswerte angeben. Desweiteren ist es möglich für die Hintergründe von Elementen Farbwerte oder auch Bilder zu verwenden (z.B. Texturen oder Bilder)

<code>color</code>	<code>#ff00ff</code> (Hexwert)
<code>background-color</code>	<code>rgb(37, 72, 0)</code>
<code>border-color</code>	<code>cadetblue</code> (Farbname)



**Beispiel**

***03\_RWD/04\_colors/***

## QUELLEN:

► [https://www.w3schools.com/cssref/css\\_colors.asp](https://www.w3schools.com/cssref/css_colors.asp)

# SCHRIFTEN UND WEBFONTS

Auf Websites lassen sich auf verschiedene Arten eigene Schriften einbinden. In der Regel will man ja nicht die Systemschriften verwenden, da diese im Grunde nie zum CI / CD des jeweiligen Kunden passen oder auch anderen gestalterischen Aspekten nicht genügen. Es gibt zu einem Plattformen wie „Google Fonts“ oder „Typekit“ welche ein sehr großes Portfolio an Schriften bereitstellt, zum anderen auch Websites wie Behance, Fontsquirrel oder Dafont, welche Schriften kostenlos zum Download anbieten.



**Beispiel**

***03\_RWD/05\_fonts/***

## QUELLEN:

- ▶ <https://fonts.google.com/>
- ▶ <https://typekit.com/>
- ▶ <https://www.fontsquirrel.com/>
- ▶ <http://www.dafont.com/de/>

# FLOATING UND POSITIONIERUNG

Floating dient der Manipulation des regulären Seitenflusses, so kann man z.B. dafür sorgen, dass nicht nur „inline“-Elemente wie Bilder, Links oder generell Texte nebeneinander dargestellt werden, sondern auch „block“-Elemente, welche standardmäßig ja einen Umbruch verursachen (z.B. <div>, <p>, <h1>, usw.). Allerdings bringt ein Float „Nebenwirkungen“ mit sich, welcher man sich bewusst sein sollte, da durch einen Float der reguläre Seitenfluss verändert wird. Sprich nachfolgende HTML Elemente versuchen dann dieses „gefloate“ Element zu umfließen, was zu unschönen Ergebnissen führen kann. Aber auch dafür gibt es Lösungsansätze:



## Beispiel

***03\_RWD/06\_floating/***

## QUELLEN:

► [https://www.w3schools.com/css/css\\_float.asp](https://www.w3schools.com/css/css_float.asp)



# INTERAKTIONEN, EFFEKTE, ANIMATIONEN

Richtig lebendig wird eine Website erst dann, wenn diese auch auf den Nutzer reagiert. In der simpelsten Form kann das z.B. ein Link sein, welcher die Farbe ändert sobald man mit der Maus darüberfährt oder auch ein Bild, welches sich bei Berührung vergrößert. Animationen ermöglichen so visuelles Feedback, welches dem Benutzer Rückmeldung gibt und somit die Usability der Website unterstützt.



**Beispiel**

***03\_RWD/07\_effects/***

## QUELLEN:

► <https://tympanus.net/codrops/>

# LOGO / NAVIGATION

Das Logo und die Navigation sind unerlässliche Elemente der eigenen Website und diese werden z.B. mit einem Bild oder einer Grafik, bzw. in Form einer Liste in die Website integriert.

In unserem Fall:

- ▶ Home
- ▶ Portrait
- ▶ Lost-Place
- ▶ Über mich



***Beispiel***

***03\_RWD/08\_navigation/***

# FOOTER

Auch der Footer ist ein wichtiger Bereich der Website, da er z.B. Kontaktdaten, Social Media Profile oder Auflistungen von Beiträgen enthalten kann. Generell dient er als „Überbringer“ von knappen Informationen. Häufig sind dort z.B. auch die Navigationspunkte Impressum / Datenschutz untergebracht, da diese für deutsche Seitenbetreiber essenziell sind.



**Beispiel**

**03\_RWD/09\_footer/**

# ***RESPONSIVE WEBDESIGN PRAXISTIPPS***



# GRUNDLAGEN

**Beim Responsive Webdesign (im Deutschen auch responsives Webdesign genannt oder kurz RWD) handelt es sich um ein gestalterisches und technisches Paradigma zur Erstellung von Websites, so dass diese auf Eigenschaften des jeweils benutzten Endgeräts, vor allem Smartphones und Tabletcomputer, reagieren können.**

Der grafische Aufbau einer „responsiven“ Website erfolgt anhand der Anforderungen des jeweiligen Gerätes, mit dem die Site betrachtet wird. Dies betrifft insbesondere die Anordnung und Darstellung einzelner Elemente, wie Navigationen, Seitenspalten und Texte, aber auch die Nutzung unterschiedlicher Eingabemethoden von Maus (klicken, überfahren) oder Touchscreen (tippen, wischen). Technische Basis hierfür sind die neueren Webstandards HTML5, CSS3 (hier insbesondere die Media Queries) und JavaScript.

## QUELLEN:

► [https://de.wikipedia.org/wiki/Responsive\\_Webdesign](https://de.wikipedia.org/wiki/Responsive_Webdesign)

# KLASSISCHER UND SKALIERTER VIEWPORT

Der Viewport ist der Bereich in dem eine Website oder Anwendung auf dem jeweiligen Gerät dargestellt wird. Also auf einem Smartphone ist das z.B. der gesamte Bildschirm, auf einem Desktop PC z.B. kann das auch nur der Bereich sein, den das Browserfenster einnimmt.

Hierbei kann man nun bestimmen, ob eine Website immer in den gerade verfügbaren Viewport angepasst werden soll (kennt man von vielen Websites, welche nicht responsive sind --> sehr kleine Ansicht und nicht optimiert) oder ob eine Anpassung oder Skalierung unterbunden werden soll, was dazu führt, dass Media Queries auch wirklich greifen können (--> RWD). Darüber hinaus lassen sich auch noch Zoomstufen einstellen oder inwieweit der Nutzer, z.B. durch „Pinch and Zoom“ die Seite vergrößern darf.



**Beispiel**

***03\_RWD/11\_viewport/***

## QUELLEN:

► [https://wiki.selfhtml.org/wiki/HTML/Kopfdaten/meta#Viewport\\_einstellen](https://wiki.selfhtml.org/wiki/HTML/Kopfdaten/meta#Viewport_einstellen)

# BREAKPOINTS & MEDIA QUERIES

Media Queries ist eine Methode um in CSS zu definieren, dass bei bestimmten Bedingungen andere Regeln gelten sollen wie vorher. Das kann unter anderem das Erreichen eines Breakpoints (z.B. 480 Pixel Breite) sein oder, dass die Website mit einem Retina Display aufgerufen wird, welches eine doppelte Pixeldichte wie ein Desktop Monitor aufweist. Denkbar wären auch spezielle Regeln für den Druck.

Beim Responsive Webdesign werden diese Media Queries primär dazu verwenden um die Layouts einer Website auf die verschiedenen Displaygrößen anzupassen und zu optimieren indem man Eigenschaften wie Breite & Höhe von Elementen verändert.



## **Beispiel**

***03\_RWD/12\_media-queries/***

## QUELLEN:

► [https://www.w3schools.com/css/css\\_rwd\\_mediaqueries.asp](https://www.w3schools.com/css/css_rwd_mediaqueries.asp)

# ADAPTIVE LAYOUT & RESPONSIVE LAYOUT

## ADAPTIVE = ANPASSUNGSFÄHIG

Adaptive Websites nutzen Media Queries dazu um spezielle Anpassungen für bestimmte Geräte oder Displaygrößen zu erzeugen, wie z.B. kleine Monitore, Tablets oder Smartphones. Die Anpassungen hierbei sind häufig sehr speziell und sind z.B. nur für ein iPhone optimal.

## RESPONSIVE = REAKTIONSFÄHIG

Responsive Websites basieren auf einem „fluiden“ Raster und haben durch Media Queries die Fähigkeit das Design und die Inhalte auf verschiedenen Displaygrößen und Geräten optimal darzustellen, da nicht für spezielle Viewports optimiert, sondern der zur Verfügung stehende Platz optimal genutzt wird.

## QUELLEN:

- <https://blog.kulturbanause.de/2012/11/adaptive-website-vs-responsive-website/>



# FRAMEWORKS: BOOTSTRAP, FOUNDATION & CO.

## FOUNDATION

- ▶ sehr flexibel
- ▶ responsive
- ▶ mobile First Approach
- ▶ relativ wenige Designvorgabe, eher technische und strukturelle Basics
- ▶ extrem flexibel und individualisierbar
- ▶ SASS

## QUELLEN:

- ▶ <http://foundation.zurb.com/>
- ▶ <http://getbootstrap.com/>
- ▶ <https://purecss.io/>

## BOOTSTRAP

- ▶ schnelles Prototyping möglich
- ▶ adaptive
- ▶ viele vordefinierte Komponenten
- ▶ Große Verbreitung / Community
- ▶ Less / SASS



# »MOBILE FIRST« & »CONTENT FIRST

Mit dem Begriff „**Mobile First**“ wird ein Konzept für das Webdesign sowie die Konzeption von Websites bezeichnet. Dieses Konzept sieht vor, dass die für mobile Endgeräte optimierte Version zuerst entsteht und sukzessive Erweiterungen stattfinden.

„**Content First**“ geht hier noch einen Schritt weiter, denn bevor überhaupt etwas technisch umgesetzt wird, macht man sich Gedanken darüber, welcher Inhalt in welcher Form und welchem Medium kommuniziert werden soll (Text oder Bild, Video oder Audio, usw.) Erst dann wenn das klar ist, beginnt man das Design auf den Content anzupassen.

## QUELLEN:

- <http://bradfrost.com/blog/post/atomic-web-design/>

# ***FLEXIBLE GESTALTUNGSRASTER (GRIDS) ENTWICKELN***

Grid-Systeme sind der Schlüssel um viele die meisten Inhalte schon einmal grob in ein responsives Raster einzuteilen, welches sich an die Gegebenheiten der zahlreichen Geräte anpasst, sodass die optimale Darstellung des Inhalts gewährleistet bleibt.



***Beispiel***

***03\_RWD/15\_flexible-grids/***

## **QUELLEN:**

► <http://bradfrost.com/blog/post/atomic-web-design/>

# EIN VERÄNDERTES BOX-MODELL

Reponsive Layouts sind darauf angewiesen, dass man die Größen von Elementen relativ zum Viewport definieren kann. Dies lässt sich z.B. mit einer prozentualen Größenangabe leicht umsetzen. Probleme tauchen hierbei auf, wenn man bei mehrspaltigen Layouts mit Abständen und Rändern arbeiten möchte, da diese zu mit der prozentualen Breite addiert werden und hierbei Gesamtbreiten von über 100% herauskommen, was zu einem Bruch der Spalten führt. Dies lässt sich aber mit einer Anpassung des Box-Modells lösen.



**Beispiel**

***03\_RWD/15\_flexible-grids/***

## QUELLEN:

► <http://bradfrost.com/blog/post/atomic-web-design/>

# GRAFIKEN IM RESPONSIVE WEBDESIGN

Eine neue Herausforderung stellt sich auch bei den Bildern, da diese ja nun auch für verschiedenste Bildauflösungen optimiert werden müssen, denn ein Smartphone benötigt keine 2000x2000 Pixel Bilder, welche z.B. ein Desktop PC ausreizen könnte. Ein Weg wäre, Bilder einfach immer so groß abzulegen, dass genug Pixel vorhanden sind um jede erdenkliche Auflösung abdecken zu können. Aber das führt selbst bei guter Komprimierung zu echt schweren Websites, was sich vor allem bei Smartphones, in Form von miesen Ladezeiten oder enormen Verbrauch von Datenvolumina niederschlägt. Intelligent lässt sich das mit sog. „srcset“ Attributen lösen, da der Browser so selbst anhand des Gerätes und der zur Verfügung gestellten Bildgrößen entscheiden kann, welche Version des Bildes benötigt wird. Einfluss auf die Bildwahl haben zu einem Größe des Viewports, aber auch die „Pixel Density, bzw. die Display-Ratio“ des jeweiligen Gerätes. Man könnte sogar noch weiter gehen und dem Browser mitteilen, ob das Bild die volle Breite oder nur einen Teil des Viewports einnehmen soll (z.B. mehrspaltige Darstellung).



**Beispiel**

**03\_RWD/18\_srcset/**

## QUELLEN:

► <https://blog.kulturbanause.de/2014/09/responsive-images-srcset-sizes-adaptive/>

# VIDEOS IM RESPONSIVE WEBDESIGN

Videos müssen etwas anders behandelt werden wie Bilder, da man diese in der Regel nicht einfach beschneiden oder neu rendern kann, was auch enorme Rechenkapazitäten voraussetzen würde. Deswegen bedient man sich eines kleinen Tricks um eine Skalierung von eingebundenen Videos zu ermöglichen, dabei reicht es aus welches Verhältnis das Video aufweist (z.B. 16:9 Format).



**Beispiel**

***03\_RWD/19\_rwd-video/***

# TABELLEN IM RESPONSIVE WEBDESIGN

Tabellen sind generell schon sehr flexibel, da sich die Zellen verschiedenen Breiten anpassen. Allerdings ist es häufig bei mobilen Geräten so, dass man mehrere Spalten einfach nicht unterbringen kann. Hier löst man das Problem so, dass die Tabelle um 90 Grad gedreht wird, sodass man von oben nach unten und nicht von links nach rechts lesen muss, da man ja nach unten beliebig Platz hat. Es gibt auch Lösungen, welche Javascript verwenden um die Struktur der Tabelle komplett zu verändern oder um ein Scrollen innerhalb der Tabelle zu ermöglichen. Hier sollte man immer abhängig vom Inhalt entscheiden was die beste Lösung ist.



**Beispiel**

***03\_RWD/20\_rwd-table/***

## QUELLEN:

► <https://blog.kulturbanause.de/2012/06/tabellen-im-responsive-webdesign/>

# JAVASCRIPT

**JavaScript** (kurz JS) ist eine Skriptsprache, die ursprünglich 1995 von Netscape für dynamisches HTML in Webbrowsern entwickelt wurde, um Benutzerinteraktionen auszuwerten, Inhalte zu verändern, nachzuladen oder zu generieren und so die Möglichkeiten von HTML und CSS zu erweitern.

## QUELLEN:

- <https://developer.mozilla.org/de/docs/Web/JavaScript>





# BILDSLIDER / CAROUSEL



**Beispiel**

**03\_RWD/22\_slider/**

## QUELLEN:

- ▶ <https://jquery.com/>
- ▶ <https://jquery.com/> <http://kenwheeler.github.io/slick/>



# IMAGE-FEED



**Beispiel**

***03\_RWD/23\_image-feed/***

## QUELLEN:

- ▶ <https://jquery.com/>
- ▶ <https://jquery.com/> <http://kenwheeler.github.io/slick/>

