Responsive Web Design

© Udo Schöfer

DATEV eG

www.github.com/udos86

A Dao of Flexibility

"Today, anything that's fixed and unresponsive isn't web design, it's something else.

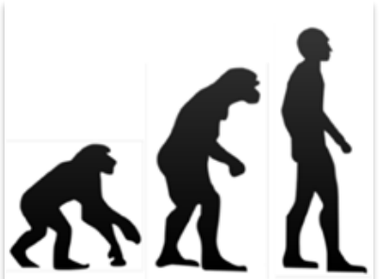
If you don't embrace the inherent fluidity of the web, you're not a web designer, you're something else."

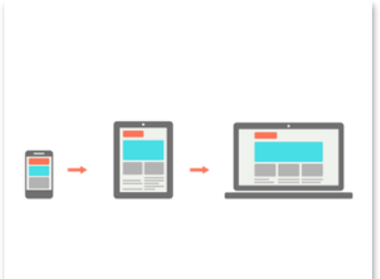
– Andy Clarke –

Modern Web Development


HTML5


CSS3

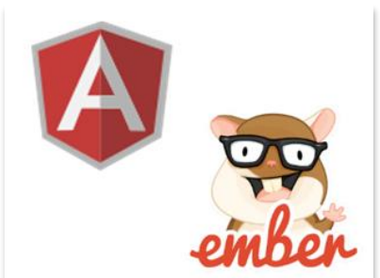

Progressive Enhancement


Mobile First

Meta Viewport


Responsive Web Design

ECMAScript 2015/16



Client-Side MV*

Isomorphic JavaScript

Build Automation

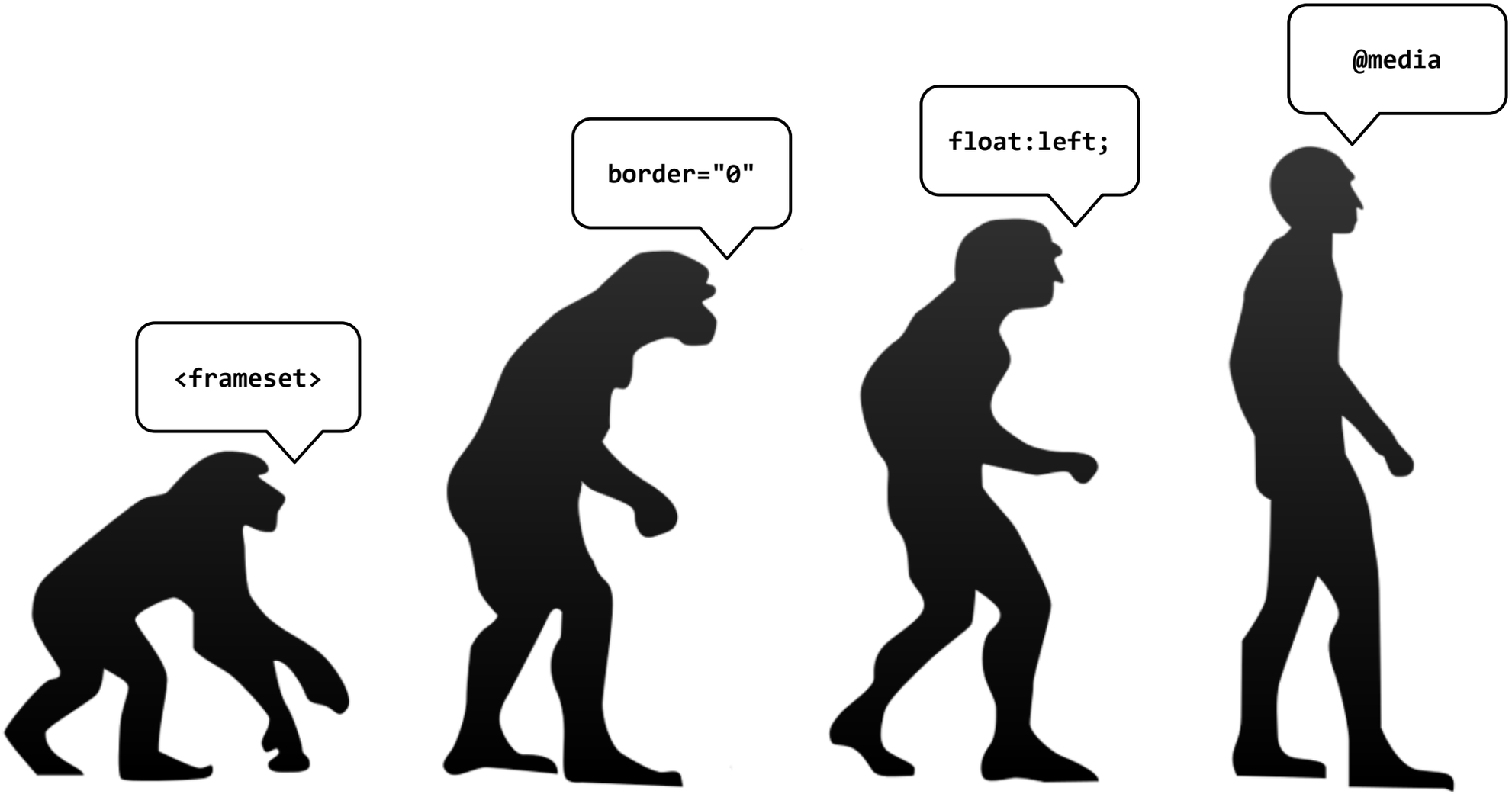

Test-Driven Development


Responsive Images

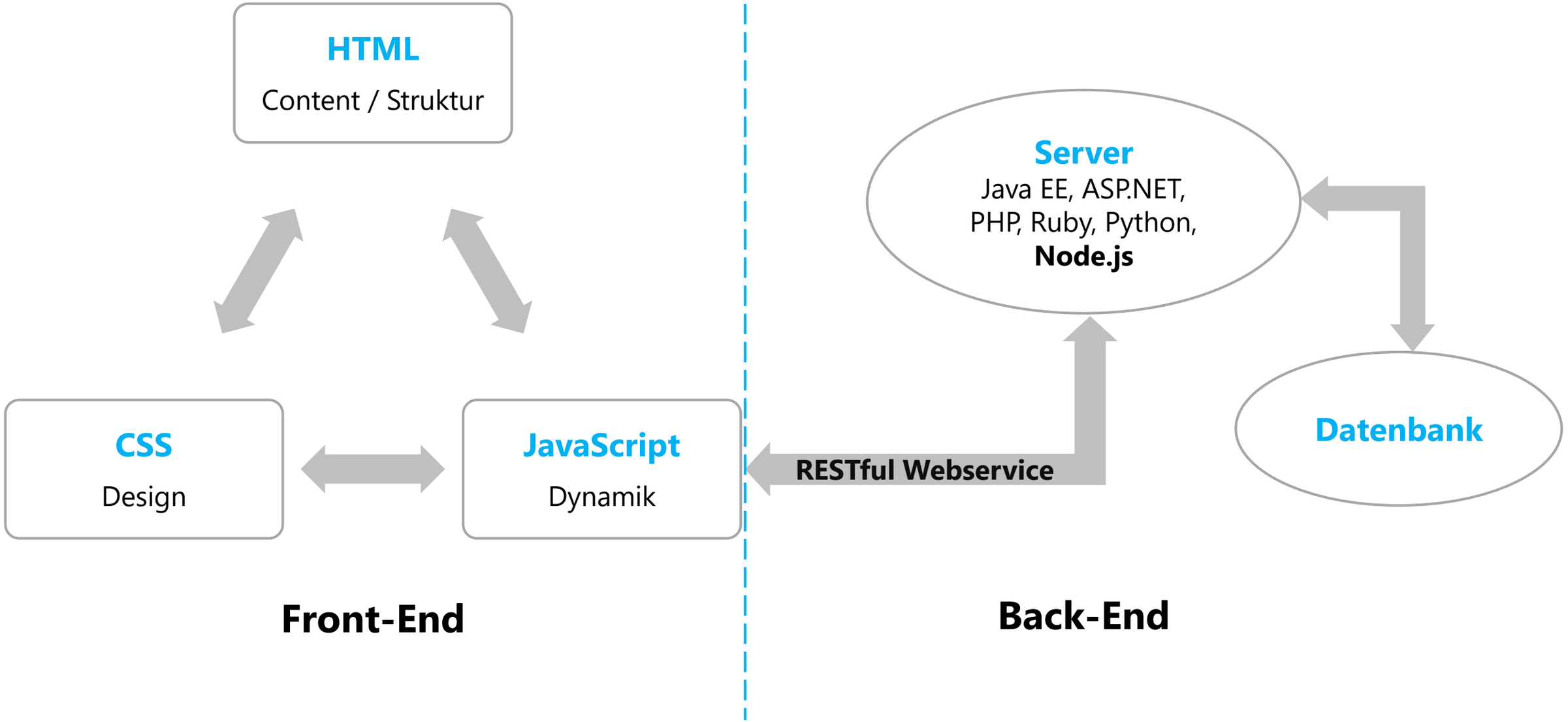

Web Components


Progressive Web Apps

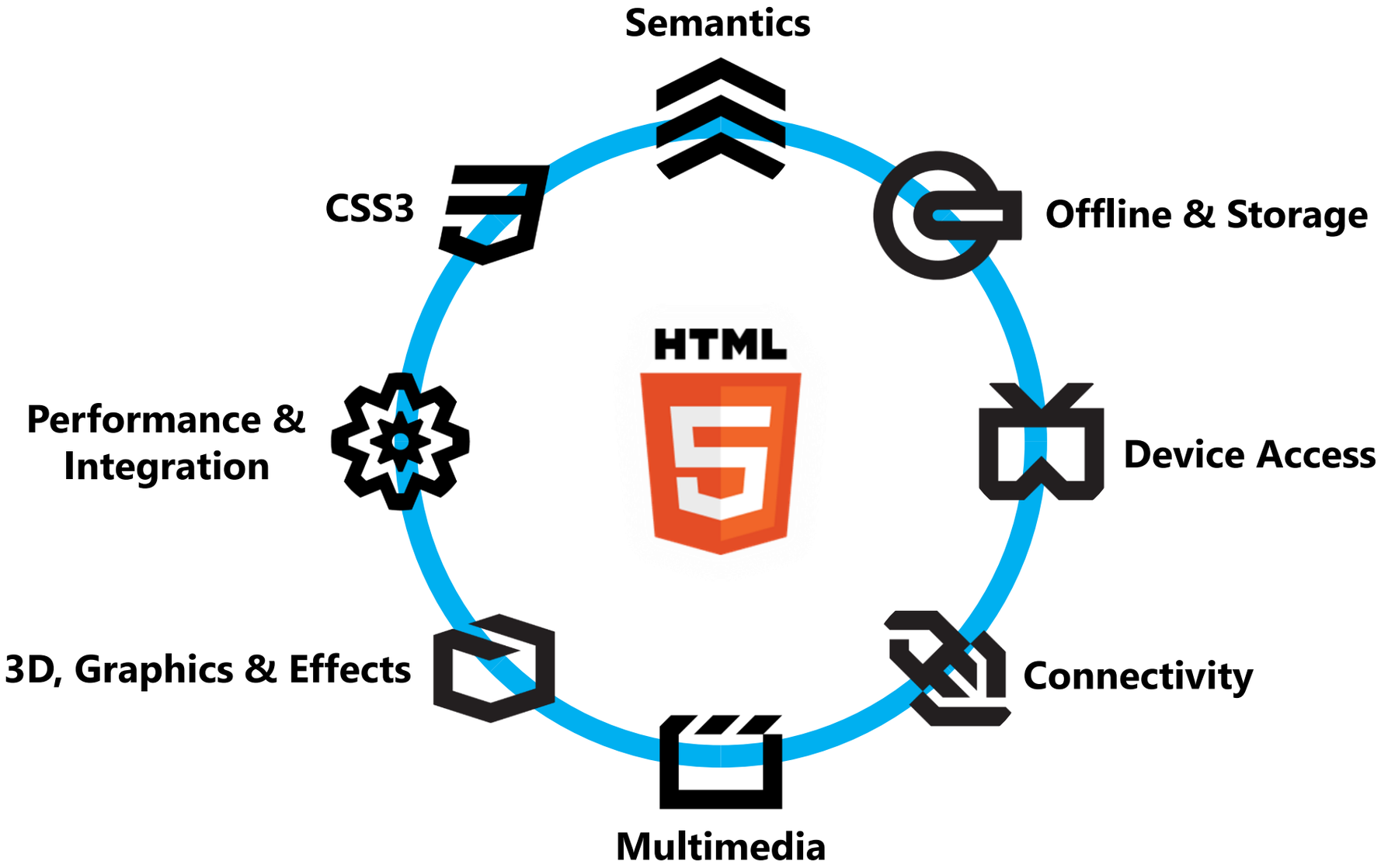
Web Design Evolution



Client-Server-Architektur



HTML5


Semantics
CSS3
Offline & Storage
HTML
Performance & Integration
Device Access
3D, Graphics & Effects
Connectivity
Multimedia

CSS3

Compositing and Blending 1.0

Backgrounds and Borders Module Level 3

Box Alignment Module Level 3

Transforms

Text Level 3

Fonts Module Level 3

Flexible Box Layout Module

Grid Layout

Speech Module

Multi-column Layout Module

Values and Units Module Level 3

Basic User Interface Module Level 3

Transitions

Ruby Module

Color Module Level 3

Variables Module Level 1

Regions Module Level 3

Fragmentation Module Level 3

Selectors Level 3

Animations

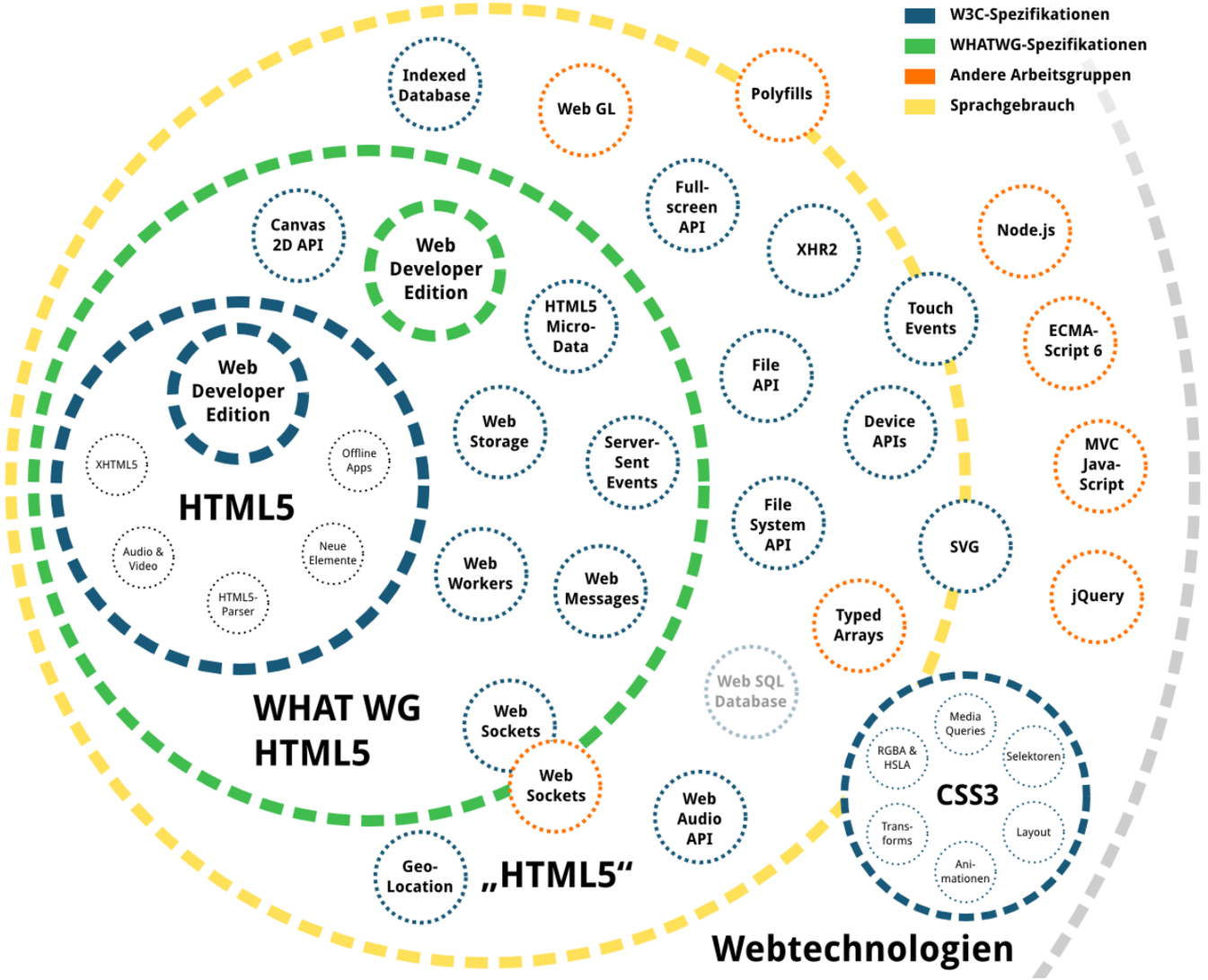
Exclusions and Shapes Module Level 3

Media Queries

Image Values and Replaced Content Module Level 3

CSS3

Neue Webtechnologien*



* nach Peter Kröner

Progressive Enhancement

Progressive Enhancement

- Gegenentwurf zu Graceful Degredation (Regressive Enhancement)
- Idealtypische Webdesign-Strategie
- Stete Zugänglichkeit des Content unabhängig von der Leistungsfähigkeit der Browserumgebung
- Sukzessive Steigerung der User Experience für moderne Browser ausgehend von robuster Code-Basis
- Nutzung von serverseitigem Fallback-Rendering, Offline-Mechanismen, Feature Detection (und Polyfills)
- Manifestiert in **Progressive Web Apps**
- ***„The web is not a platform. It's a continuum."*** (Jeremy Keith)

Mobile First

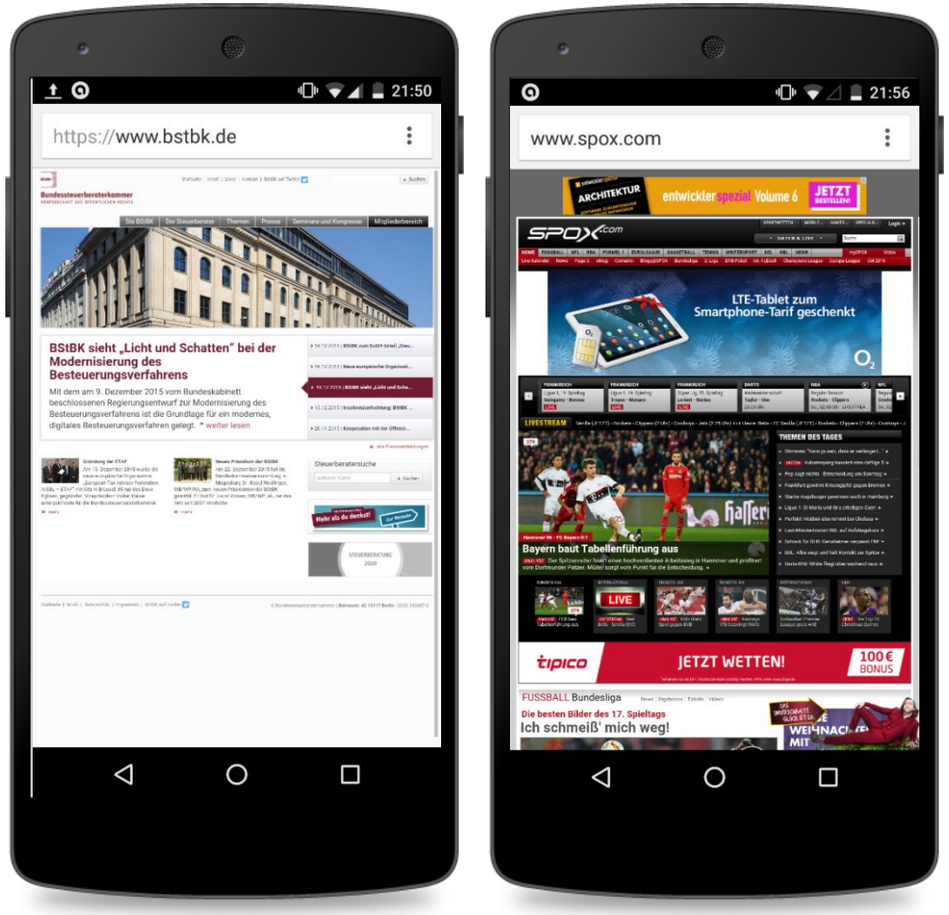
- Design-Philosophie: „***Web applications should be designed for mobile first!*** "
- Postuliert von Luke Wroblewski (2009)
- Weiterführung bzw. Bestandteil von „Progressive Enhancement“
- Gestaltung ausgehend von minimalen Bildschirmabmessungen
- Mehrere ausschlaggebende Faktoren
  - *„Mobile is exploding“*
  - *„Mobile forces you to focus“*
  - *„Mobile extends your capabilities“*

Device Context Continuum

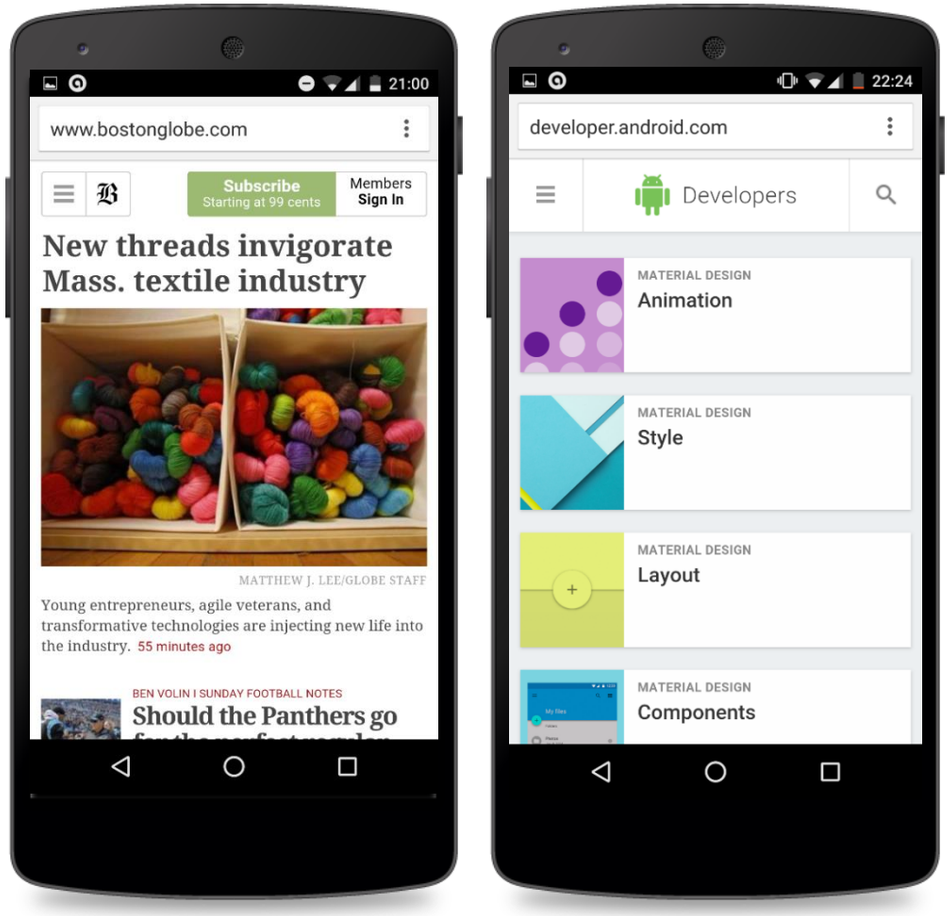
Responsive Web Design

- Erdacht von Ethan Marcotte (2010)
- „State-of-the-Art“
- Flexibles und geräteunabhängiges Webdesign
- Bildschirmgerechte Reaktivität des User Interface
- Kombination verschiedener Techniken
  - **Flexible Grids**
  - **Flexible Images / Media**
  - **Media Queries**

Praxisbeispiele Web Design



Fixed Design

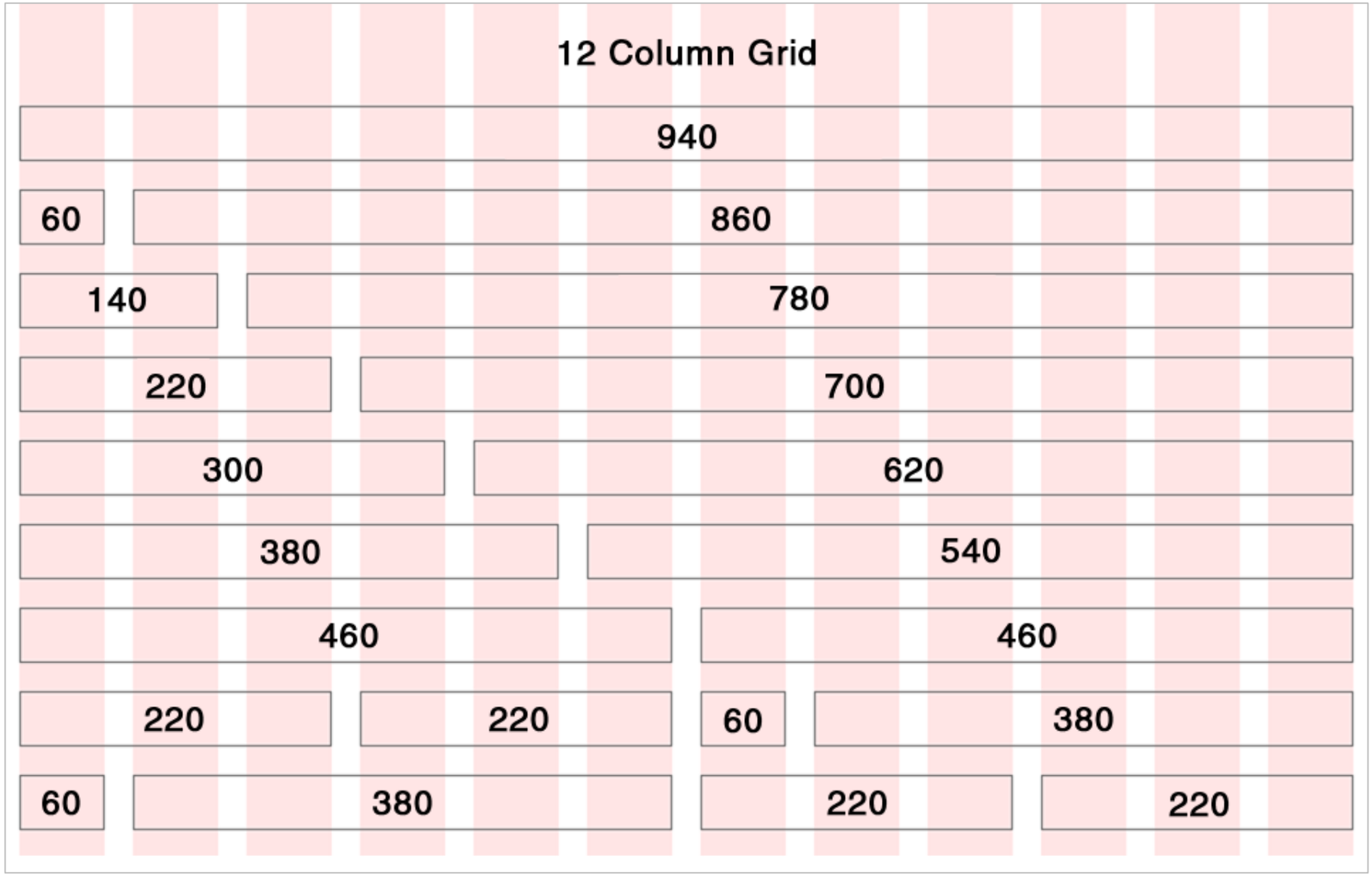


Responsive / Adaptive Design

CSS Grundwissen

- **Selektoren**: Auswahl von Elementen für Style-Anweisungen
- **Spezifität**: Gewichtung eines Selektors
- **Box Model**: Zusammensetzung von Breite und Höhe eines Containers
- **Relative Einheiten**
- **`display`**: Box-Verhalten eines Elements (Inline, Block-Level, Flex, etc.)
- **`position`**: Positionierung eines Elements im Seitenfluss (`static`, `relative`, `absolute`, `fixed`)
- **`float`**: Umfließen von Elementen

Grid System


12 Column Grid
940
60
860
140
780
220
700
300
620
380
540
460
460
220
220
60
380
60
380
220
220

Flexible Grid

- Keine statisch fixierte Grid-Breite (wie z.B. `width: 960px`)
- Verwendung relativer Einheitengrößen
- Auszeichnung von `width`, `padding`, `margin` und `font-size` als Proportion des übergeordneten („*containing*“) Elements
- Prozentuale Umrechnung absoluter Pixel-Werte der Designvorlage
- Einfache Formel zur Konvertierung: $\boldsymbol{result} = \frac{\boldsymbol{target}}{\boldsymbol{context}} * \mathbf{100}\%$

CSS Grid Frameworks

Name	Responsive Grid	Responsive Flexbox Grid	Website
960 Grid System	✘	✘	http://960.gs/
Bootstrap	✓	✓	http://getbootstrap.com/css/#grid
css-wizardry-grids	✓	✘	http://csswizardry.com/csswizardry-grids/
Cute Grids	✓	✘	http://www.cutegrids.com/
Foundation	✓	✓	http://foundation.zurb.com/grid.html
Neat	✓	✘	http://neat.bourbon.io/
Responsive Grid System	✓	✘	http://www.responsivegridsystem.com/
Simple Grid	✓	✘	http://thisisdallas.github.io/Simple-Grid/
Skeleton	✓	✘	http://getskeleton.com/
Unsemantic	✓	✘	http://unsemantic.com/
CSS Grid Layout Module Level 1 (W3C Working Draft)			http://www.w3.org/TR/css-grid-1/

Display-Kenngrößen

- **Diagonale**: Abstand zweier diagonal gegenüber liegenden Bildschirmecken in absoluten Maßeinheiten (Zoll), z.B. 17", 24", usw.
- **Seitenverhältnis**: Verhältnis der absoluten Breite des Bildschirms zur Höhe, z.B. 16:9, 16:10, usw.
- **Auflösung**: Die Gesamtanzahl physischer Pixel eines Displays, z.B. 320 x 480, 1280 x 720, 1920 x 1080, usw.
- **Punktdichte**: Die Anzahl physischer Pixel bezogen auf die Länge eines Inchs, z.B. 120dpi, 160dpi, 240dpi, usw.
- **Orientierung**: Die Ausrichtung des Geräts, d.h. hochkant (Portrait) oder quer (Landscape)

Absolute CSS-Längeneinheiten

Bezeichnung	Symbol	Verfügbar in		Entsprechende Pixelgröße
		CSS2.1	CSS3	
Zentimeter	**cm**	✓	✓	1 cm ~ 37.795 px
Millimeter	**mm**	✓	✓	1 mm ~ 3.780 px
Inch (Zoll)	**in**	✓	✓	1 in. ~ 96 px
Punkt	**pt**	✓	✓	1 pt ~ 1.333 px
Pica	**pc**	✓	✓	1 pc ~ 16 px

- Physische Originalgrößen nur für Druckausgabe relevant
- Keine Berücksichtigung displayspezifischer Basisgröße DPI
- Pixel-Abbildung auf Basis historischer DPI-Norm 96dpi
  → 1 CSS Inch entspricht 96 CSS-Pixeln
- Sonderfall: Pixel werden in CSS als relative Einheit betrachtet

Pixels

- Grundsätzliche Unterscheidung von drei Pixel-Arten
  - **Device-Pixel**: Echtes physische Pixel des Endgeräts
  - **CSS-Pixel**: In CSS-Deklarationen verwendete Pixeleinheit
  - **Device-Independent Pixel (Dip)**: Zusätzliche Abstraktion für hochauflösende Displays
- Die Größe eines CSS-Pixels ist flexibel skalierbar
- In CSS definierte Pixel sind nicht zwangsläufig identisch mit physischen Pixeln
- Die Anzahl an Dips ist identisch mit der Anzahl an CSS-Pixeln, welche zur optimalen Betrachtung bei 100% Zoom nötig sind

Relative CSS-Längeneinheiten

Bezeichnung	Symbol	Verfügbar in		Relativ zu
		CSS2.1	CSS3	
Pixel	**px**	✓	✓	Bildschirmauflösung
Prozent	**%**	✓	✓	Container
Em	**em**	✓	✓	Schriftgröße
X-height	**ex**	✓	✓	Schriftgröße des Containers
Breite des Viewports	**vw**	✘	✓	Viewport
Höhe des Viewports	**vh**	✘	✓	Viewport
Kleinere Viewportgröße	**min**	✘	✓	Viewport
Breite von 0	**ch**	✘	✓	Schriftart des Containers
m-Höhe des Root-Elements	**rem**	✘	✓	Schriftart des Wurzelelements
Grid	**gd**	✘	✓	Text-Grid eines Containers

Flexible Images / Media

- Grundlegende CSS-Regel für flexibel skalierende Medieninhalte:

  ```
  img, embed, object, video { max-width: 100%; }
  ```

- Limitierung der maximalen Bildausmaße auf die Grenzen des umschließenden Elternelements

- Effekt: Flexible Skalierung bei Veränderung der Viewport-Größe

- Problematik: Dateien werden stets in voller Größe geladen, d.h. kritischer Overhead bei geringer Bandbreite → Abhilfe durch Responsive Images

CSS Media Queries (1)

- CSS2: **Media Types**
  - Bereitstellung separater Stylesheets für verschiedene Ausgabemedien (Drucker, Screen-Reader, etc.)
  - `<link rel="stylesheet" type="text/css" href="print.css" media="print">`
- CSS3: **Media Queries**
  - Erweiterung von Media Types um kontextspezifische Abfragemöglichkeiten von **Media Features**
  - Höhe und Breite des Geräts (`device-width` | `device-height`)
  - Höhe und Breite des Viewports (`width` | `height`)
  - Orientierung des Geräts (`portrait` | `landscape`)
  - Und weitere (z.B. `resolution`, `device-aspect-ratio`, etc.)

CSS Media Queries (2)

- Kombination von Media Features durch logische Konnektoren: **,** (***Comma***), **and**, **not**
- Eingrenzung von Wertebereichen via **min-** und **max**-Präfix
- Externe Zuordnung durch `media`-Attribut des `link`-Tags:

```
<link rel="stylesheet" type="text/css" media="only screen and (max-device-width:
480px)" href="small-device.css" />
```

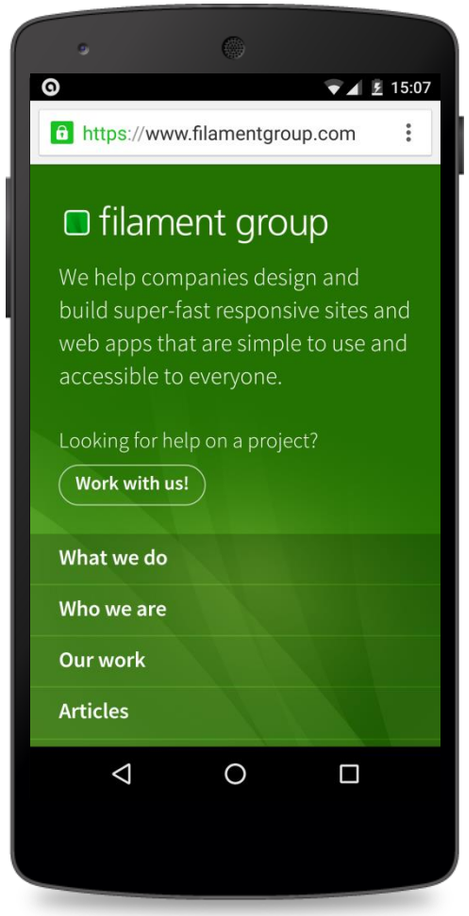
- Direkte Implementierung über `@media`-Regel in Stylesheet:

```
@media all and (min-width: 48em) and (orientation: landscape) {
      body {font-size: 95%;}
}
```

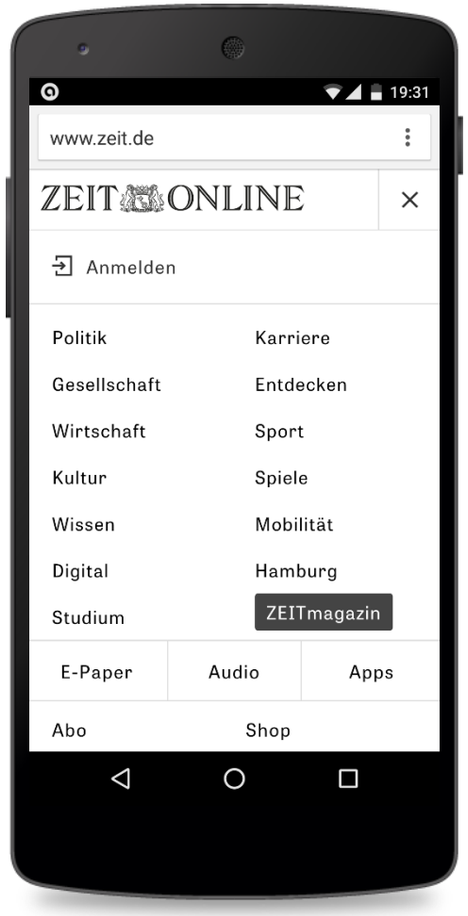
Responsive Navigation

- Entwurf responsiver Navigation erfordert hohe Akribie
  - Grundsätzlich keine ideale User Experience für schmale Viewports
  - Klassische Dropdown-Menüs für Touch-Screens nicht geeignet (kein hover-Zustand)
  - Gewährleistung einfacher Erreichbarkeit trotz großer Scroll-Höhe
- Viele **Design Patterns**, welche stets Stärken und Schwächen aufweisen, z.B.
  - Stacked Top Links
  - Toggle / Overlay
  - Flyout / Drawer / Off-Canvas
- Initial häufig verborgen und oftmals per ≡ („Hamburger“)-Button einblendbar
- Breadcrumbs für komplexe Site-Strukturen empfehlenswert

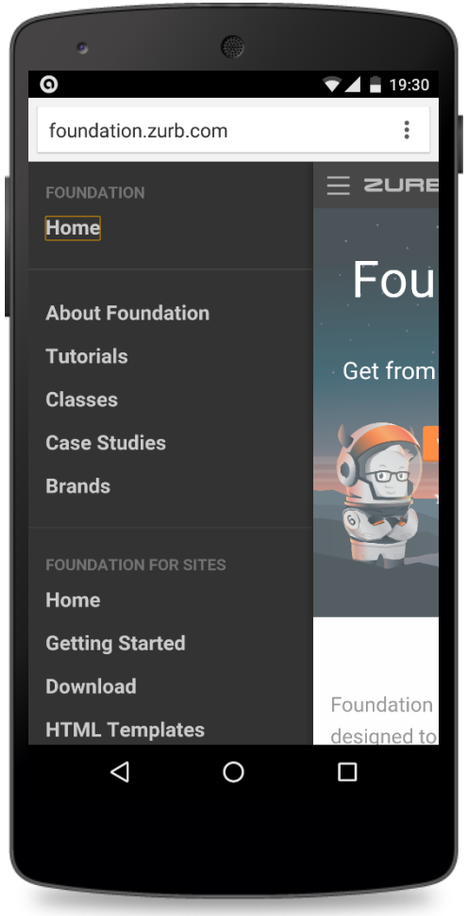
Praxisbeispiele Responsive Navigation



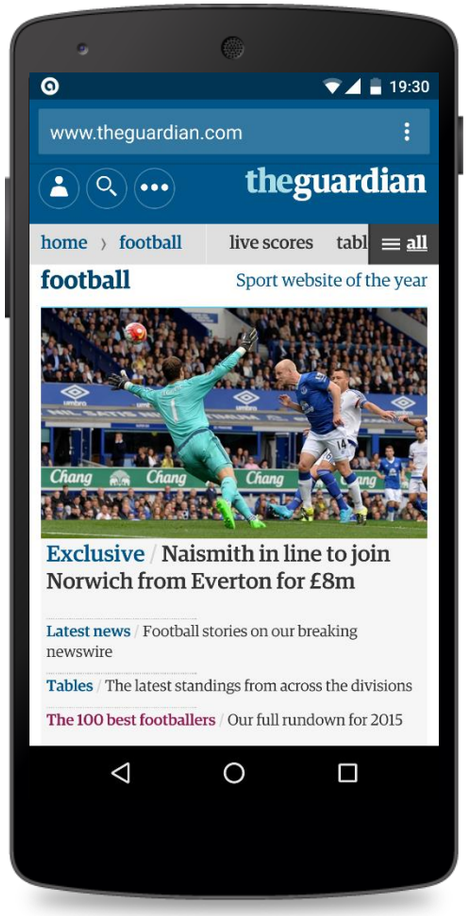
Stacked Links



Toggle



Off-Canvas Drawer



Horizontal Overflow

Mobile Browser

Mobile Browser

- Starke Fragmentierung der mobilen Browserlandschaft (vor allem im Android-Universum)
- Großteil mobiler Browser basieren auf WebKit-Bibliothek
- Dennoch: „***There is no WebKit on mobile***“ (Peter-Paul Koch)
- Extrem unterschiedliche Leistungsfähigkeit
- Speziell abgewandeltes Viewport-Konzept und Rendering-Verfahren

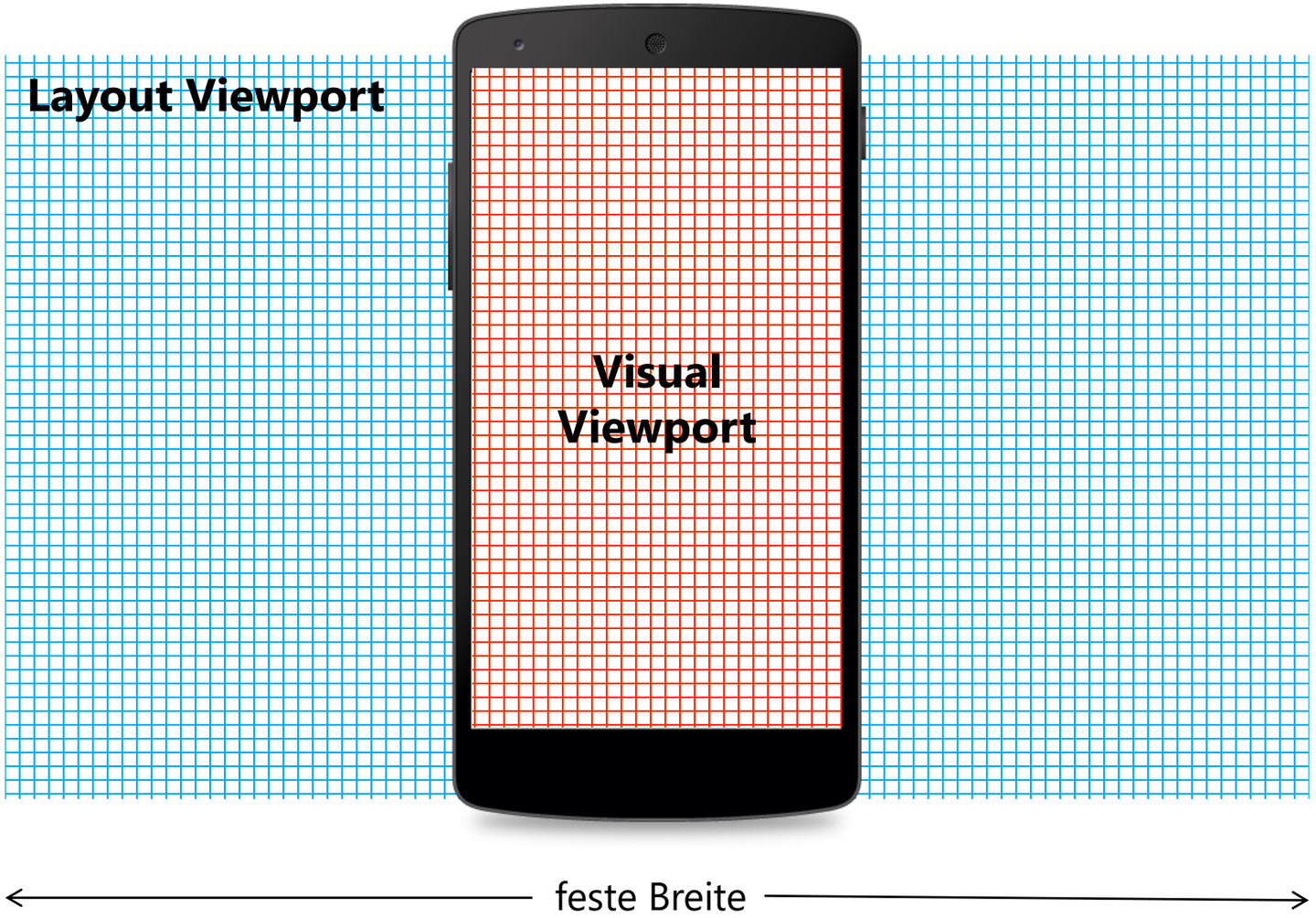
Mobile Viewports (1)

- Desktop: Viewport ≙ Browser-Fenster
- Mobile: Orientierung und Ausmaße des Displays häufig verschieden zu Widescreen-Monitoren
- Problematik: Typisches Desktop-Rendering würde auf mobilen Endgeräten (nicht optimierte) Layouts extrem stauchen
- Lösung: Trennung des Viewport mobiler Browser in
  - echten, sichtbaren **Visual Viewport**
  - breiteren, virtuellen **Layout Viewport**
  - optimal dimensionierten Ideal Viewport

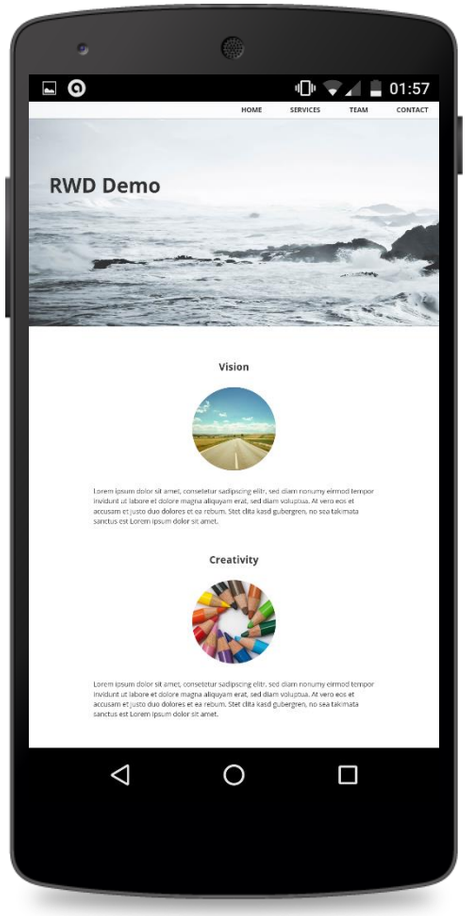
Mobile Viewports (2)

- Layout Viewport
  - Virtueller Viewport-Rahmen mit standardmäßig fester Breite
  - Größer bemessen als tatsächlich sichtbarer Viewport des mobilen Endgeräts
  - Basis auf der sämtliche CSS-Deklarationen interpretiert werden
- Visual Viewport
  - Fenster zur Darstellung eines Ausschnitts des Layout Viewports
  - Frei beeinflussbar durch den Anwender per Scrollen (Drag-Geste) und Zoomen (Pinch-Geste)

Mobile Viewports (3)



Mobile Viewports (4)

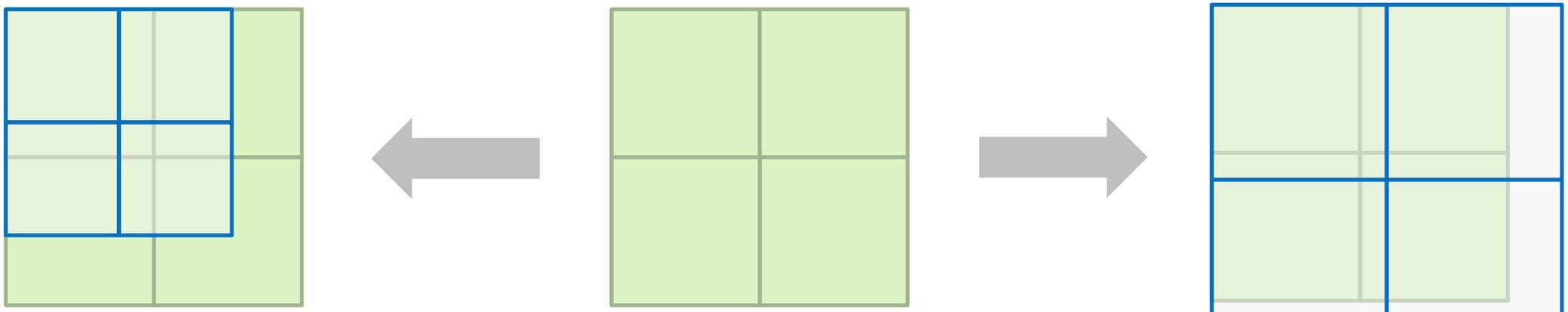


Standard (Zoom-Out): Visual Viewport ≙ Layout Viewport

Desktop Zooming (1)

- Desktop-Viewport wird durch Zooming reduziert / erweitert
- Größe jedes CSS-Pixels nimmt prozentual zu / ab
- Gleiche Anzahl an Device-Pixeln enthält in der Folge weniger / mehr CSS-Pixel
- Relativ deklarierte Eigenschaften eines Elements werden neu berechnet, z.B. `margin-left: 10%;`

Desktop Zooming (2)



Zoom-Out

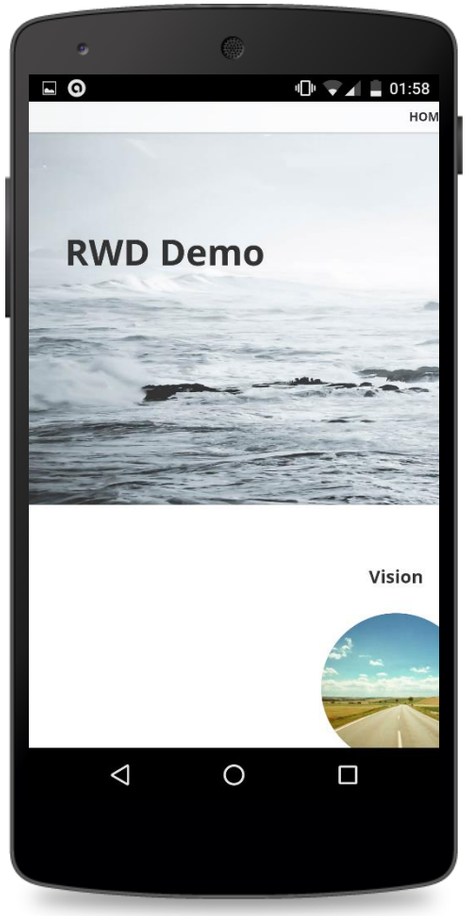
Zoom: **100%**

Zoom-In

Mobile Zooming (1)

- Gänzlich andersartiges Zooming-Verfahren in mobilen Browsern
- Dimensionen des Visual Viewports werden reduziert / erweitert
- Layout Viewport bleibt in **statischem** Zustand, d. h. CSS-Deklarationen werden nicht neu berechnet
- Visual Viewport enthält de facto weniger / mehr CSS-Pixel
- Bei Orientierungswechsel Anpassung an Änderungen des Visual Viewports durch automatisches Zooming des Browsers

Mobile Zooming (2)



Zoom-In: Visual Viewport < Layout Viewport

Meta Viewport (1)

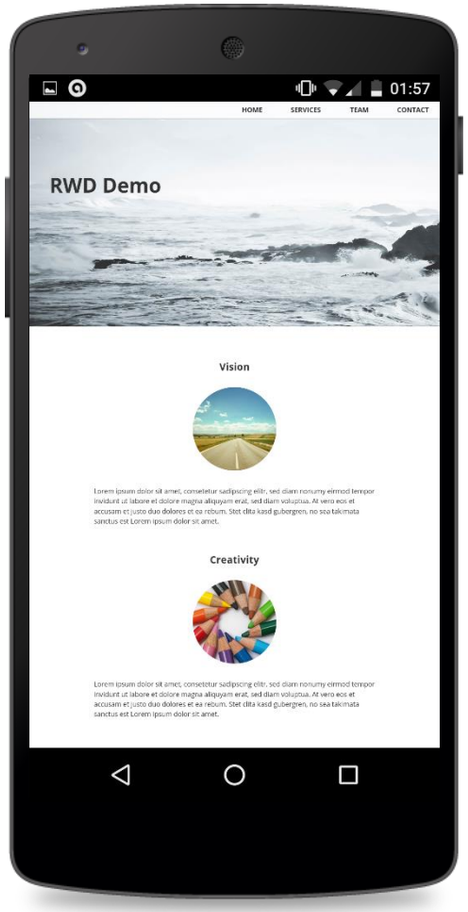
- Manuelle Festlegung des Layout Viewports im HTML-head möglich
- Basis-Syntax: **`<meta name="viewport" content="...">`**
- Verschiedene Eigenschaften des `content`-Attributs, z.B.
  - `width` | `height`: Breite / Höhe des Layout Viewports
  - `initial-scale`: Zoom-Level beim initialen Laden
  - `maximum-scale` | `minimum-scale`: Maximaler / Minimaler Zoomfaktor
  - `user-scalable`: Anwender darf / darf nicht zoomen

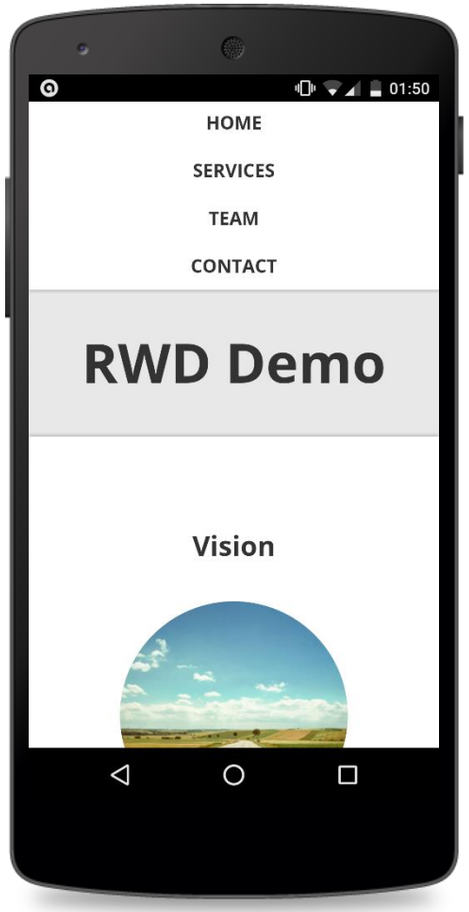
Meta Viewport (2)

- Primäres Attribut für mobile Site-Optimierung: `width`
- Best Practice für sämtliche responsiv gestalteten Webinhalte:

  `<meta name="viewport" content="width=device-width, initial-scale=1">`
- Ausdruck `width=device-width` weißt Layout Viewport die **ideale** Breite für das jeweilige Display (in Device-Independent-Pixels) zu → Ideal Viewport
- `width`-Attribut erhält durch `width=device-width` sinnvolle Größenzuweisung zur Weiterverwendung in Media Queries
- Empfehlung: In Media Queries stets `width` verwenden (nicht `device-width`)

Meta Viewport (3)





ohne Meta Viewport

`width = device-width`

Device-Independent Pixels

- Moderne, hochauflösende Displays („*Retina*“) weisen sehr hohe DPI-Werte auf
- Wert von `device-width` müsste theoretisch 1:1 anwachsen, d.h. optimale Site-Darstellung wäre nicht mehr gewährleistet
- Praxis: Mobile Browser antizipieren idealen Standardwert für `device-width` → **Ideal Viewport**
- Wert von `device-width` entspricht nicht zwingend der echten Display-Breite, sondern basiert auf **Device-Independent-Pixels** (dips), z.B. identisch für iOS Safari Non-Retina und Retina: 320
- Browser kompensiert hohe Punktdichten durch Vervielfachung von Größenangaben, z.B. `20px` in iOS Safari Non-Retina entspricht `40px` in iOS Safari Retina → **Device-Pixel Ratio**

Device-Pixel Ratio

- Kenngrößen-Definition: **Anzahl physischer Pixel pro Device-Independent Pixel**
- Verhältnis > 1 bei hohen physischen Auflösungen, z.B. iOS Safari Non-Retina = 1 und Retina = 2
- Auswertung von Device-Pixel Ratio in Media Queries und JavaScript möglich
  - `(min-device-pixel-ratio: 2),(-webkit-min-device-pixel-ratio: 2)`
  - `(min-resolution: 2dppx), (min-resolution: 192dpi)`
  - `window.devicePixelRatio`
- Wichtige Eigenschaft zur Bereitstellung hochauflösender Bildinhalte für Retina-Displays
  - Bereitstellung von Grafiken in doppelter Auflösung via CSS Media Query und gleichzeitiges „Downsampling“
  - Verwendung spezieller JavaScript-Bibliotheken (z.B. retina.js)
  - Responsive Images

Viewport-Übersicht*

Konzept	Beschreibung	Media Query	Meta-Viewport	JavaScript
Physischer Screen	Anzahl physischer Pixel des Geräte-Displays	`device-width` (urspr. Definition)	-	`screen.width` (urspr. Definition)
Layout Viewport	Initialer CSS-Inhaltsbereich (oft deckungsgleich mit Ideal Viewport)	`width` `height`	`width`	`document.documentElement.clientWidth` `document.documentElement.clientHeight`
Visual Viewport	Momentane Größe des sichtbaren Site-Bereichs	-	-	`window.innerWidth` `window.innerHeight`
Ideal Viewport	Ideale Ausmaße des Layout Viewport für optimale Darstellung	`device-width` `device-height`	`width=device-width`	-
Resolution	Verhältnis zwischen physischer Screen-Größe und Ausmaß des Ideal Viewport	`resolution` `device-pixel-ratio`	-	`window.devicePixelRatio`
Orientation	Momentane Ausrichtung des Geräts: Portrait oder Landscape	`orientation`	-	`window.orientation`
Zooming	Momentaner Zoomfaktor der Site relativ zum Ideal Viewport	-	`*-scale`	-
Scrolling	Momentane Position des Virtual Viewport relativ zum Layout Viewport	-	-	`window.pageXOffset` `window.pageYOffset`

* nach PPK

Responsive Images (1)

Responsive Images (2)

- Unterschiedliche Anwendungsfälle für flexible Bildinhalte
  - Alternative Motivausschnitte abhängig von Größe / Orientierung des Viewport („***Art Direction***“)
  - Alternative Auflösungen abhängig von Pixeldichte des Ausgabebildschirms („***Resolution Switching***“)
  - Ausmaße in relativer Abhängigkeit zur Größe des Viewport
  - Alternative Dateiformate zur Optimierung von Ladezeiten
- Auf klassischem Wege ohne JavaScript per Markup nicht implementierbar
  - Preloading-Mechanismus des Browser für Bilder setzt unmittelbar bei Seitenaufruf ein
  - `<img>`-Tag kann per `src`-Attribut nur eine einzige Bildquelle zugewiesen werden

Responsive Images (3)

Eigenschaft	Entwickler zur Implementierzeit bekannt	Browser zur Ladezeit bekannt
Größe des Viewports des Ausgabe-Browser	✘	✓
Größe des Bildes relativ zum Viewport	✓	✘
Device-Pixel Ratio des Ausgabebildschirms	✘	✓
Größe der originalen Bilddateien	✓	✘

→ Notwendigkeit einer HTML5-Spezifikationserweiterung: **Responsive Images**

Responsive Images (4)

- Präzisierung des Darstellungskontexts durch neue Attribute für `<img>`-Tag
  - **`srcset`**: Komma-getrennte Auflistung von Dateipfaden mit Density oder `width`-Deskriptor
  - **`sizes`**: Komma-getrennte Liste von Media Conditions und relativen Größenangaben
  - **`media`**: Angabe des Gültigkeitsbereichs durch validen Media Query
  - **`type`**: MIME-Type-Angabe alternativer Dateiformate
- Einführung des **`<picture>`-** und **`<source>`-**Elements
  - Definition alternativer Bilddateien
  - Deklarative Beeinflussung des Ladens der jeweils adäquaten Ressource
- Browser evaluiert Deskriptoren und lädt eigenständig die passendste Ressource

Responsive Images (5)

Eigenschaft	Entwickler zur Implementierzeit bekannt	Browser zur Ladezeit bekannt
Größe des Viewports des Ausgabe-Browser	✘	✓
Größe des Bildes relativ zum Viewport	✓	`sizes`
Device-Pixel Ratio des Ausgabebildschirms	✘	✓
Größe der originalen Bilddateien	✓	`srcset`

Responsive Images in der Praxis (1)

```
<img srcset="image.jpg, image-2x.jpg 2x" src="image.jpg" alt="a simple image">
```

Density-Deskriptoren ausschließlich bei fixen Bildgrößen verwenden!

```
<img srcset="image-large.jpg 1240w, image-medium.jpg 620w, image-small.jpg 310w"
     sizes="(min-width: 41.25em) 38.75em, calc(100vw - 2.5em)"
     src="image_medium.jpg"
     alt="An image showing a beautiful landscape.">
```

Für flexibel skalierende Bilder immer `sizes`-Attribut in Kombination mit `width`-Deskriptoren in `srcset` verwenden!

`img`-Tags reichen für den Großteil aller Anwendungsfälle von Responsive Images aus!

Responsive Images in der Praxis (2)

```
<picture>
        <source media="(min-width: 900px)" srcset="image-vertical.jpg">
        <source media="(min-width: 750px)" srcset="image-horizontal.jpg">
        <img src="image-fallback.jpg" alt="just another image">
</picture>
```

`media`-Attribut gibt Browser konkrete Vorgabe und im Gegensatz zu `srcset` und `sizes` keinen Auswahlspielraum!

```
<picture>
        <source type="image/svg+xml" srcset="logo.xml">
        <source type="image/webp" srcset="logo.webp">
        <img src="logo.png" alt="logo">
</picture>
```

CSS Architektur (1)

- Komplexität großer Stylesheets wird zu oft unterschätzt
  - Gefahr hoher Redundanz durch signifikante CSS-Sprachgrenzen
  - Globale Gültigkeit verursacht undurchsichtige Style-Anweisungen
- Wichtige Anforderung an Stylesheets: **Skalierbarkeit**
  - **Vorhersehbarkeit**: Verwendung von Style-Anweisungen sollte nur zu erwarteten Auswirkungen führen und keine Seiteneffekte in anderen Site-Bereichen verursachen
  - **Wiederverwendbarkeit**: Abstrahierung von CSS-Regeln, so dass neue Komponenten auf Basis existierender Style-Anweisungen realisiert werden können
  - **Wartbarkeit**: Hinzufügen neuer Komponenten und Regel sollte nicht im Refactoring bereits existierenden CSS resultieren

CSS Architektur (2)

- Typische Schwachstellen in Stylesheets
  - Undurchsichtige Spezifitäten durch zu willkürliche Selektoren: `#main-nav ul li ul li a {…}`
  - Modifizierung von Komponenten anhand von Elternelementen: `#sidebar .my-widget { width: 200px }`
  - Zu generische Klassennamen: `.my-widget .title {…}` | `.my-widget .content {…}`
  - Zu viele Anweisungen innerhalb eines Selektors
- Best Practices
  - Verwendung flacher Klassenstrukturen und Verzicht auf `id`-Selektoren
  - Einsatz bewährter Design Patterns: **BEM**, **Atomic CSS**, **OOCSS**, **SMACSS**, **SUITCSS**, etc.
  - Namespacing von CSS-Klassen, z.B. `.company-component {…}`
  - Verwendung von **CSS-Präprozessoren**

BEM

- Konzipiert bei Yandex
- Populäre Methode zur Benennung und Notation von CSS-Klassen
- Zusammensetzung von Komponenten bzw. Stylesheets aus drei Kategorien
  - **Block**: Top-Level-Abstraktion einer Komponente, z.B `.button { ... }`
  - **Element**: Semantisch gebundenes Kind einer Komponente, z.B. `.button__label { ... }`
  - **Modifier**: Styling einer Komponente oder eines Elements, z.B. `.button--large { ... }`

Atomic CSS

- Konzipiert bei Yahoo
- Klassenbasiertes Architekturmuster für Stylesheets
- Nur eine einzelne Style-Deklaration pro CSS-Klasse („*Single purpose styling unit*“)
- Funktionale Notation von Klassennamen, z.B. `W(50%)` oder `Bgc(#0280ae.5)`
- Styling von Komponenten durch Zuweisung verschiedenster CSS-Klassen via HTML-Markup:

```
<div class="IbBox W(50%) Ta(c) Bgc(#0280ae.5) C(#fff)">Box-1</div>
```

Vendor Prefixes

- Proprietäre Integration experimenteller CSS-Eigenschaften durch Browser-Hersteller lange Zeit per vendor prefix: `-webkit-`, `-moz-`, `-o-`, `-ms-`
- Berücksichtigung sämtlicher Herstellerpräfixe und präfixloser Standard-Notation für maximale Kompatibilität notwendig
- Von Web Designern in Stylesheets oftmals unsauber implementiert
- In Web-Community kontrovers diskutiert
- Mittlerweile abgelöst durch Aktivierung neuer, noch nicht standardisierter Features per Flags in Browser-Einstellungen

CSS Präprozessoren

- Erweiterung von CSS um zusätzliche Dynamik durch **Metasprachen**, z.B.
  - Variablen: Speicherung global einsetzbarer Werte
  - Nesting: Verschachtelung von Selektoren
  - Mixins: Mehrfache Verwendung eines gesamten Styling-Blocks
  - Funktionen: Dynamische Berechung von Werten und Erstellung von Selektoren
  - u.v.m.
- Standardlösungen: **SASS** und **Less**
- Kompilierung in im Browser lauffähiges CSS

Touch Events

- Spezielles Ereignismodell für Touch-Interaktion
  - `touchstart`
  - `touchmove`
  - `touchend`
  - `touchcancel`
- Parallele Simulation von `mouse`-Events aus Kompatibilitätsgründen
- `click`-Event ≠ `touchstart`–Event: Zeitverzögertes Auslösen des simulierten Klick-Ereignisses um ca. 300ms zur Berücksichtigung von Doppelklicks
- Entwurf normalisierter **Pointer-Events** durch Microsoft → W3C Recommendation

Mobile Testing

- Hoher Testaufwand für responsive Webinhalte durch extreme Hardware-Vielfalt
- Einplanung von Budget für physische Testgeräte notwendig
- Direktes Debugging auf mobilem Endgerät nicht praktikabel
- Unterschiedliche Verfahren zur Inspektion und Fehleranalyse
  - **Mobile Emulation**
  - **Remote Debugging**
  - **Synchronised Cross-Browser Testing**
  - **Performance Testing**

Mobile Emulation

- Softwareseitige Imitation einer mobilen Browserumgebung
- Simulation von Auflösungen, Meta Viewport, Device-Pixel Ratio, Touch-Gesten, usw.
- Visuelle Übersicht zu Breakpoints aus Media Queries
- In modernen Browser-Entwicklerwerkzeugen unterschiedlich umfangreich integriert
  - Chrome / Opera: „Device Mode“
  - OSX Safari: „Responsive Design Mode“
  - Firefox / Firefox Developer Edition
  - Internet Explorer 11 / Edge
- Ersetzt nicht das Testen auf „echten“ Geräten

Remote Debugging

- Originale Code-Einsicht in mobilen Browser „aus der Ferne“ mit Entwicklungsrechner
- iOS: **Safari Remote Web Inspector**
  - Voraussetzungen: iOS Safari und OSX Safari, aktivierte iOS-Systemeinstellung „Web Inspector“, aktive USB-Verbindung
  - Entwicklerwerkzeuge für iOS Safari und native iOS UI/WKWebView direkt im OSX Safari aufrufbar
- Android: **Chrome Remote Debugging**
  - Voraussetzungen: ~~ADB Chrome-Extension~~, aktivierte Android-Systemeinstellung „USB-Debugging“, aktive USB-Verbindung (manuelle Installation der ADB-Treiber unter Windows notwendig)
  - Entwicklerwerkzeuge für Mobile Chrome Tabs und native Android WebView in Desktop Chrome aufrufbar
  - Live Screencasting

Synchronised Cross-Browser Testing

- Simultanes UI-Testing für unterschiedliche Displaygrößen, Browser-Versionen und Hardwarekonfigurationen in „Gerätelabor“ (privat, öffentlich oder cloud-basiert)
- Effektive Qualitätssicherung durch paralleles Spiegeln des Entwicklungsstands auf beliebige Anzahl mobiler Testgeräte
- Mehrere adäquate Werkzeuge
  - Adobe Edge Inspect CC
  - BrowserSync
  - Vanamco Ghostlab

Performance Testing

- Ladezeit für mobile Browserinhalte besonders heikel
  - Verbindungsaufbau über Mobilfunknetz grundsätzlich langsam
  - Geringe Geduld des Anwenders (maximal 10 Sekunden bis zum Abbruch des Seitenbesuchs)
- Faustregel: Kritischer (initial sichtbarer) Content sollte binnen **einer** Sekunde geladen werden
- Freie Standardwerkzeuge für detaillierte Leistungsanalyse
  - **PageSpeed Insights**
  - **WebPagetest**
- Mobile Optimierung hat positive Auswirkung auf Google-Ranking

Links (1)

- http://www.html5rocks.com
- http://www.alistapart.com
- http://www.abookapart.com/
- http://www.smashingmagazine.com/
- http://www.quirksmode.org/mobile
- https://developers.google.com/web/
- http://caniuse.com/

Links (2)

- https://wiki.selfhtml.org/wiki/Startseite
- http://www.google.com/think/multiscreen/whitepaper-sitedesign.html
- http://bradfrost.github.com/this-is-responsive
- https://css-tricks.com/
- http://responsivewebdesign.com/podcast/
- http://responsiveimages.org/
- https://www.webplatform.org/

Links (3)

- http://www.webpagetest.org/
- https://developers.google.com/speed/pagespeed/insights/
- https://developer.chrome.com/devtools
- https://www.mozilla.org/de/firefox/developer/
- http://responsivepx.com/
- http://ami.responsivedesign.is/
- http://gridpak.com/