Vorlesungsreihe
Entwicklung webbasierter Anwendungen

Entwicklung von MOBILEN Webanwendungen

Prof. Dr.-Ing. Thomas Wiedemann
email: wiedem@informatik.htw-dresden.de

HOCHSCHULE FÜR TECHNIK UND WIRTSCHAFT DRESDEN (FH)
Fachbereich Informatik/Mathematik

Gliederung

- **Aktueller Stand und neue Anforderungen**
  - Aktuelle Entwicklungen im mobilen Bereich
  - Spezielle Probleme bei mobilen Anwendungen
- **Designtechnische Lösungsansätze**
  - Basiskonzepte im Responsive Design
  - Graceful degradation vs. Progressive enhancement
  - Responsive vs. Adaptive Webdesign
- **Hardwareorientierte Lösungsansätze**
  - Datensynchronisation mit mobilen Geräten
  - Ortung und andere Geo-Funktionen
- **Ausblick**

Aktuelle Entwicklungen im mobilen Bereich

- Rasante Entwicklung mobiler Internetgeräte in Typenvielfalt und Anzahl

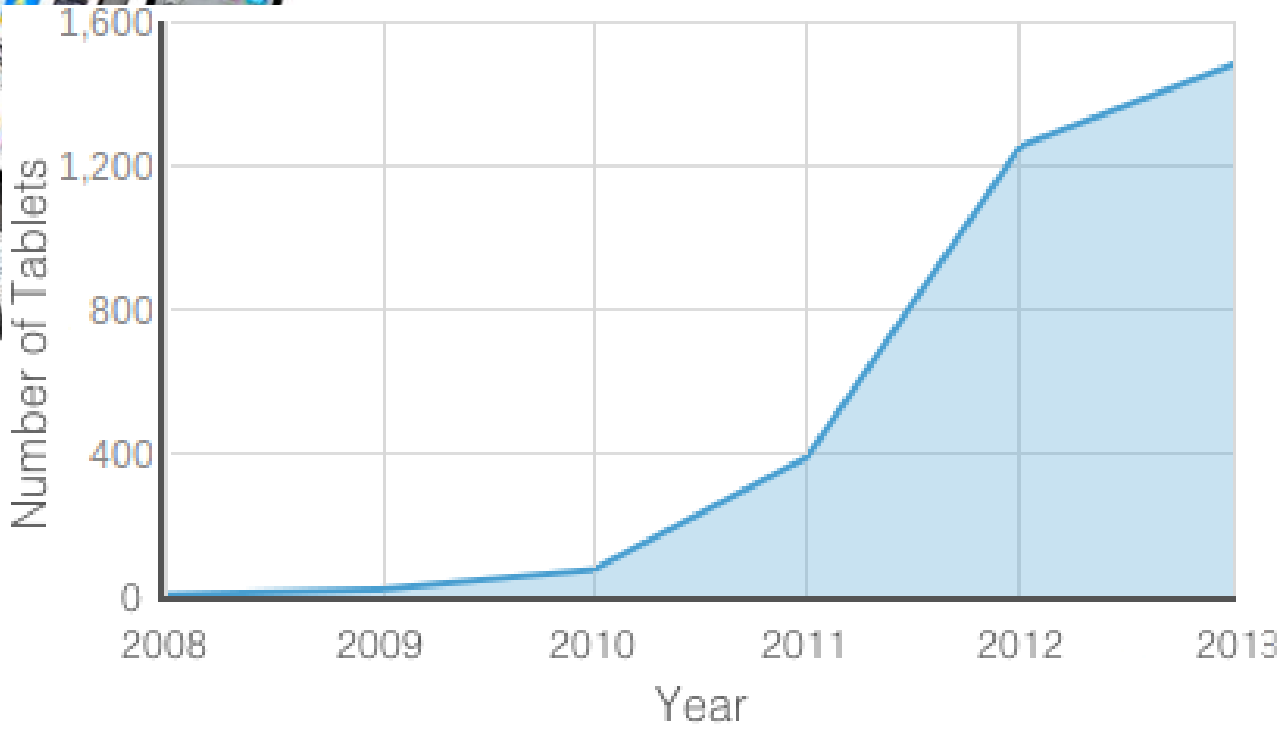


Bild-Quelle: [Mar10]

Spezielle Probleme bei mobilen Anwendungen

Geräteeigenschaften *(im Vergleich zu Desktop (PC)*

- **Anzeige**: meist deutlich kleiner als aktuelle Desktoprechner und sehr viele verschiedene Abmessungen, auch Querformate
- **Inputoptionen**: Touchscreen, tw. Spracheingabe als Standard, Bewegungssensoren (Drehen etc.),
- **System:** begrenzter Speicher, langsamerer Prozessor, dafür deutliche bessere Umweltsensorik (GPS, Kamera)
- **Netzkopplung**: relativ langsam und unsicher
- **Energie**: generell kritisch (ständige Netzverbindungen oder häufiges Polling leeren schnell den Akku)

Fazit: Aktuelle mobile Geräte weichen je nach Kriterium stark negativ wie positiv ab !

Spezifische Nutzeranforderungen im mobilen Bereich

Aktuell im Ergebnis der Geräteeigenschaften :

- **Arbeitsmodus**: meist nur im Lesemodus oder für Kommunikation und Datenaustausch (Uploads, Umfragen), Keine größeren Design- und Entwicklungsarbeiten,
- **Umgebungsbedingungen**: sehr heterogen – häufiger Ausseneinsatz mit schlechten Licht- und Bedienbedingungen (Sonnenlicht, „dicke Finger“ oder Handschuhe, Lärm)

Hauptforderung => **schnelle und einfache Bedienung**

Folge: Durch die erfolgreiche Umsetzung dieser Hauptforderung werden mobile Geräte wie Smartphones und Tabletts aktuell auch von Nutzern ohne vorherige Desktoperfahrungen genutzt.

=> **Wiederum besonders hohe Anforderungen an die Bedienfreundlichkeit und Robustheit der Anwendungen !**

Grundoptionen zur Anpassung an mobile Geräte

1. Responsive Design

- **Inhalt wird in einer Seite vom Server bereitgestellt**
- **Auf der Clientseite werden die Browserparameter getestet und die Seite möglichst optimal angepasst**

  + nur eine Seite zu entwickeln und pflegen

  - sehr hoher Anpassungsaufwand, suboptimal bzgl. Bildgrößen u.ä.

2. Adaptive Design

- **Ausgehend vom http-Request-String wird bereits auf der Serverseite eine Geräteerkennung „versucht“.**
- **Danach werden vom Server optimierte Inhalte an den Client gesendet, ggf. auch der Content in sehr unterschiedlichen Formen und Auflösungen**

  + sehr weitreichende Anpassungen möglich (ggf. vers. Webseiten)

  - sehr hoher (mehrfacher) Entwicklungsaufwand, serverseitige Geräteerkennung nicht immer einfach und korrekt !

Responsive Design – Überblick

Historie

- **Begriff geprägt im Mai 2010 von Ethan Marcotte im Artikel/Buch**
  http://alistapart.com/article/responsive-web-design/ [Mar10]

3 Kerntechnologien :

- **Flexibles, rasterbasiertes Layout**
- **Flexible Bilder und Medien,**
- **Media Queries**

Alle 3. Optionen basieren auf entsprechenden CSS-Einstellungen.

Das entsprechende Basislayout / Content muss natürlich prinzipiell entsprechend anpassbar sein oder muss angepasst werden !

Responsive Design – Flexibles Rasterlayout des Contents

Grundregeln

- **Zerlegung des Gesamtlayouts in einzelne Gridbereiche**
- **Möglichst flexible Adaption an wechselnde Geräteparameter**

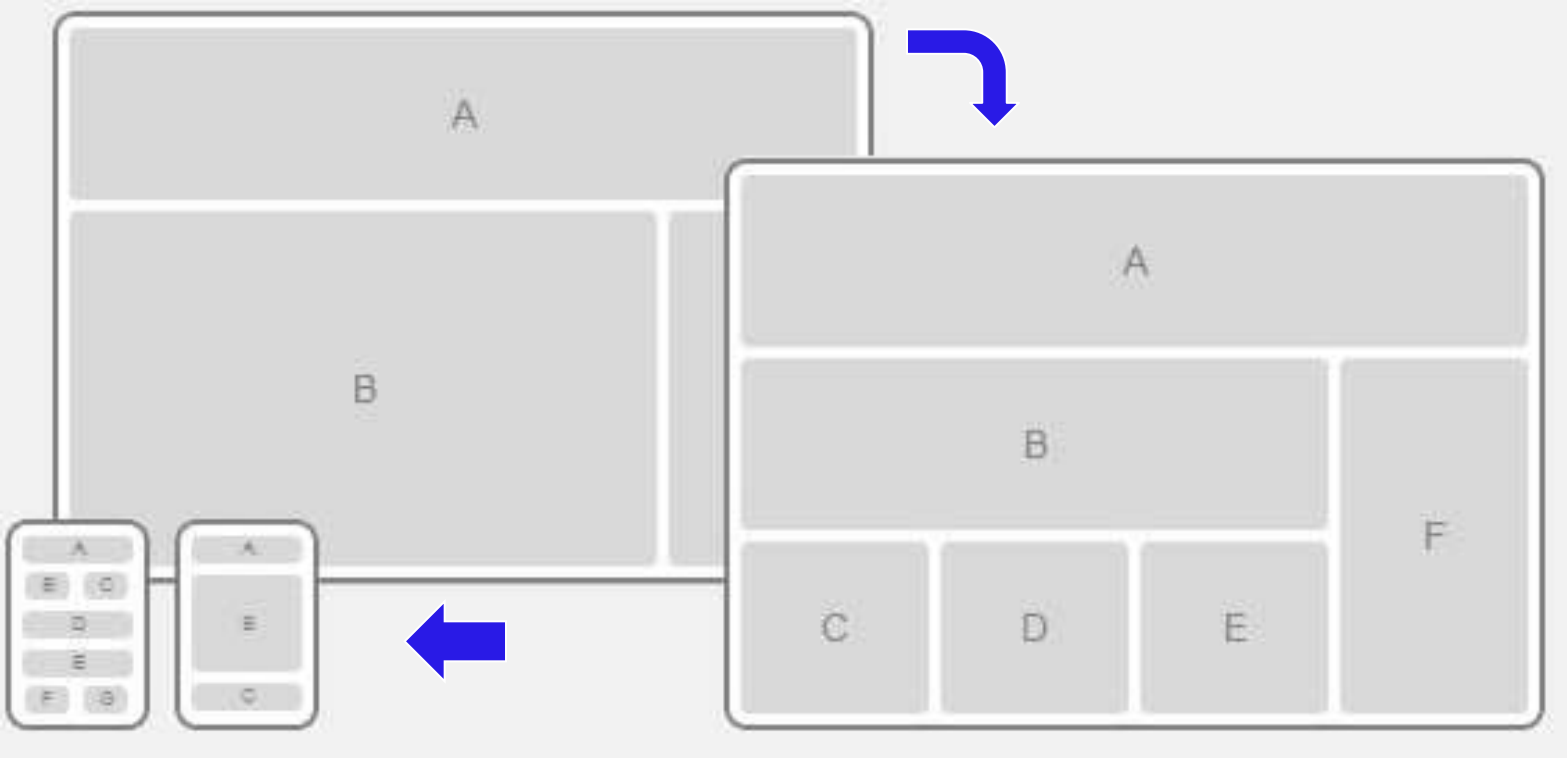
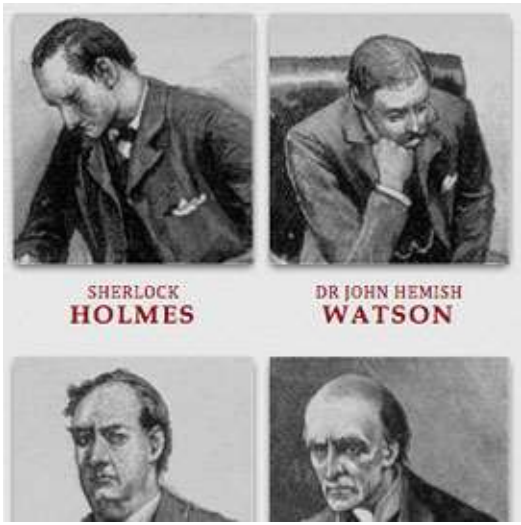


Bild-Quelle: [http://www.creativebloq.com/responsive-web-design/problems-8122790Mar10]

Responsive Design – Flexible Medien

Grundregeln

- möglichst Verzicht auf sehr große Bilder,
- ggf. Zerlegung in kleinere Einheiten und getrennte Integration mit dem Ziel eines flexiblen Umbruchs oder anderer Anpassung



CSS für flexible Medien

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

Bild-Quelle: [Mar10]

Grundregeln

- **Die beiden erstgenannten Optionen können Anpassungen in gewissen Grenzen ausführen.**
- **jedoch kritisch bei sehr großen Unterschieden wie 960px <-> 320 px**
- **Mit neuen CSS3-Optionen lassen sich sogenannte „Breakpoints“ – also Umschaltpunkte des Grundlayouts definieren**
- **Vorgehensweise analog zu alten CSS-Gerätekontexten mit Zusatzbedingung** (Optionen siehe Folgeseite):

```
<link rel="stylesheet" type="text/css"
media="screen and (max-device-width: 480px)"
href="format1.css" />
```

```
@import url(color.css) screen and (color); // in CSS-Datei
```

Nur bei POSITIVEM Test (auf alle Bedingungen) wird die angegebene CSS-Datei geladen !

Responsive Design – Media Queries - Optionen

nach W3C http://www.w3.org/TR/css3-mediaqueries/ **:**

Geräteparameter:

- **device-width, device-height -** jeweils auch mit min/max-Prefix
- **device-aspect-ratio -> ** z.B. 16/9 oder auch 1280/720
- **Device- orientation ->** portrait / landscape

Viewport bzw. Browserparameter:

- **width, height –** jeweils auch mit min/max-Prefix
- **aspect-ratio -> ->** z.B. 16/9 oder auch 1280/720
- **color ->** Anzahl Bits pro Farbe
- **color-index** bei Farb-Lookuptables -> min-color-index: 256
- **monochrome – Anzahl Graustufenwerte bei s/w**
- **Resolution - in dpi (mit min/max(**
- **Scan –** TV (“progessive”) , **grid** =1 bei reiner Textausgabe

Graceful degradation versus Progressive enhancement

Hauptprobleme bei der Anpassung sind :

- die vorhandene Browserfunktionalität (Typ / Version)
- die Funktionsfähigkeit von Javascript (und ggf. DOM / Versionspezifika.)
- die Multimediafähigkeiten (Screengröße, Auflösung, …)

Bei der Adaption gibt es zwei prinzipielle Vorgehensweisen :

Gracefule Degration

- man geht vom **Funktions-Maximum** aus
- dann wird sukzessive Ersatzfunktionalität bereit gestellt (Degration)

Progressive enhancement

- man geht vom **Funktions-Minimum** aus, was zu 100% läuft
- dann wird sukzessive neue, moderne Funktionalität bereit gestellt (Enhancement)

Graceful degradation - Beispiel

(Bsp. nach http://www.w3.org/wiki/Graceful_degradation_versus_progressive_enhancement)

- Aufbau einer Druckfunktion in der Seite:

```
<p id="printthis"> <a href="javascript:window.print()">
Print this page</a> </p>
```

- **Probleme**: Funktioniert nicht OHNE Javascript -> Lösung: danach

```
<noscript> <p class="scriptwarning"> Printing the page requires
JavaScript to be enabled. Please turn it on in your browser. </p>
</noscript>
```

- **Probleme**: Nutzer ohne Kenntnisse zur JS-Aktivierung ?
  (und es herrscht immer noch ein Gefühl einer gewissen Fehlfunktion)
- Besser: neutralerer Text mit allgemeiner Hilfefunktion :

```
<noscript> <p class="scriptwarning"> Print a copy of your confirmation.
Select the "Print" icon in your browser, or select "Print" from the
 "File" menu. </p> </noscript>
```

- **Problem**: Der erste Link existiert noch und kann auch angeklickt werden !

Progressive enhancement - Beispiel

(Bsp. nach http://www.w3.org/wiki/Graceful_degradation_versus_progressive_enhancement)

- Zu Beginn nur die allereinfachste Funktion (100% sicher):

```
<p id="printthis">Please print this page for your records.</p>
```

- Danach komplexe Javascript-Funktion mit einigen Prüfungen:

```
<script type="text/javascript">
(function(){ if(document.getElementById)
{ var pt = document.getElementById('printthis');
if(pt && typeof window.print === 'function')  // Prüfung auf Existenz „Print“ !
  { var but = document.createElement('input');  // wenn, ja Button generieren
    but.setAttribute('type','button');
    but.setAttribute('value','Print this !');
     but.onclick = function(){ window.print(); };
     pt.appendChild(but); } } })();                    // und im Form einhängen
 </script>
```

- Die gleiche Prüfung kann noch effizienter mit Jquery erfolgen.

Degradation / Progressive enhancement - Fazit

Beide Verfahren schliessen sich nicht aus und können je nach Anwendungsfall ggf. auch parallel auch angewendet werden:

- wenn möglich, ist die **„Progressive enhancement“ besser**, da keine falschen Funktionsangebote gemacht werden (wenn etwas nicht geht, ist es nicht sichtbar)
  - Neuere Funktionen können ZUSÄTZLICH angebunden werden, ohne das Gesamtsystem (einfach -> zu ganz modern) zu stören.
  - Neuer Code kann erst bei größerer Unterstützung in den verwendeten Browsern eingebaut werden – keine Notreparatur (leichtere Wartung).
  - Javascript-Code (analog zu Bsp.) erlaubt Konzentration in zentralen Libs
- **Graceful degration sinnvoll, wenn:**
  - Die gesamte Website sowieso auf Javascript und neue Funktionen setzt (z.B. sehr komplexe RIA-Anwendungen) ! Dann nur als „Absage“ !
  - Das Produkt erlaubt kein komplettes Redesign (ProEn), sondern nur leichte Ergänzungen als Addons.

Adaptive Design

Grundregeln

- **ausgehend vom http-Request-String wird bereits auf der Serverseite eine Geräteerkennung mit dem „User Agent“ –String versucht**
- **Die Auswertung erfolgt dabei meist mit größeren Datenbanken:**
- **WURFL -** „Wireless Universal Ressource File“ unter
  - **http://wurfl.sourceforge.net/ -- (tw. kostenpflichtig !)**
  - **Daten-Beispiel**

```
<device user_agent=“iphone6" actual_device_root="true"
        fall_back=“iphone_generic_series"
  <group id="product_info"> …
```

- **=> Aufwertung der Daten über PHP-Objekte (vgl. Folgeseite)**
- **oder andere Tools und Geräte-DB wie deviceatlas.com**

Adaptive Design – Gerätedetektion mit WURFL

Zentrale Einbindung der WURFL-DB

```
<?php  require_once('config-wurfl.php'); // WURFL- Konfiguration einbinden
// anfragendes Geraet ermitteln
$requestingDevice = $wurflManager->getDeviceForHttpRequest($_SERVER);
// Test ob kabelloses Geraet
$is_wireless = ($requestingDevice->getCapability('is_wireless_device') === 'true');
// Test ob Smart TV
$is_smarttv = ($requestingDevice->getCapability('is_smarttv') === 'true');
// Test ob Tablet
$is_tablet = ($requestingDevice->getCapability('is_tablet') === 'true');
// Test ob Telefon
$is_phone = ($requestingDevice->getCapability('can_assign_phone_number')
    === 'true');   ?>
```

-> danach Einbindung in die HTML- Seiten

```
<?php if ($is_phone): ?>        // HTML der Smartphone-Version
<?php elseif ($is_tablet): ?>  // HTML der Tablet-Version
```

Vergleich Responsive und Adaptive Design

Die beiden Strategien sind im Sinne der zukünftigen Weiterentwicklung eher eine **generelle Methodik** und weniger eine strenge Designvorgabe fokussieren auf sehr unterschiedliche Bereiche und müssten auch so benannt werden:

- **Client-Side Responsive Design (ClientResponDes)**
- **Server-Side-Adaptive Design (ServerAdaptDes)**

Sie schliessen sich gegenseitig nicht aus und wären auch als Hybridlösung optimal:

- **große Designunterschiede abfangen mit ServerAdaptDes**
- **finale Optimierungen und Layoutanpassungen mit ClientResponDes**

Allgemeine HTML5 / CSS /JS –Frameworks

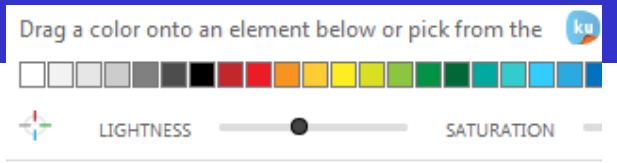
Übersicht unter

- http://designinstruct.com/roundups/html5-frameworks/
- http://www.awwwards.com/what-are-frameworks-22-best-responsive-css-frameworks-for-web-design.html
- TOP3 – Tools : **Bootstrap (by Twitter), Foundation, Skeleton**
- Vergleich der 3 Tools unter http://responsive.vermilion.com/compare.php

Allgemeine Eigenschaften der meisten HTML5/CSS-Frameworks

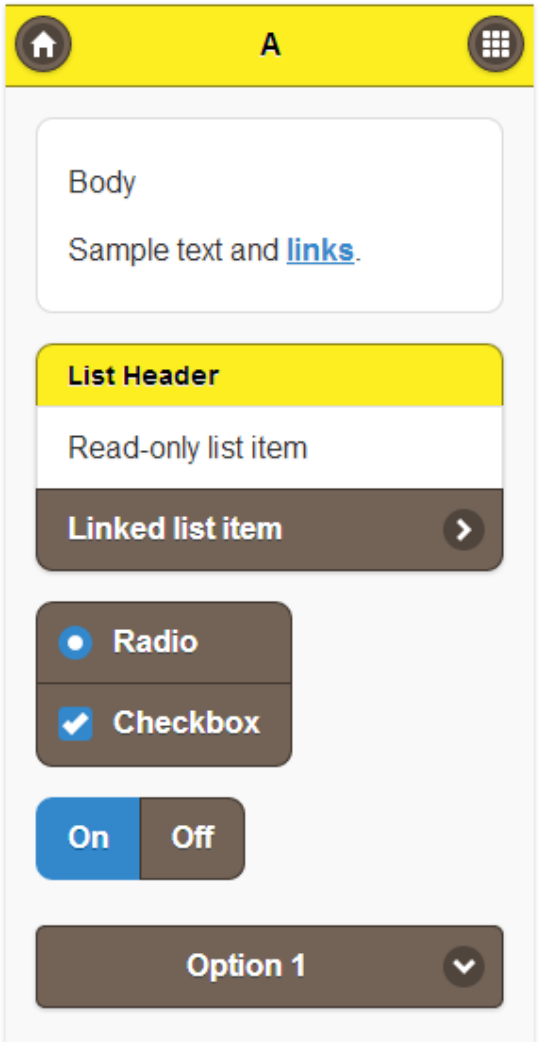
- GRID-System zur Aufteilung des Bildschirms (6 er oder 12 Grid)
- Vorbereitete Designvorlagen
- Komplexere Elemente zur GUI-Gestaltung und Formulardesign
- Tools zur Unterstützung der Layoutentwicklung (LESS-Compiler für effizientere CSS-Generierung), meist Einbindung von Jquery

Unterstützende Tools und Frameworks



JQuery-Mobile

=> jquerymobile.com



- analog zu JQuery als Framework für viele Anpassungen in Richtung mobiler Geräte
- Mit Assistenten zur Layoutgestaltung
  -> Themeroller-Module
- Vor- und Nachteil gleichzeitig
- Abhängigkeit von der korrekten Funktions-weise des Frameworks (gut bei JQ)
- Ggf. relativ gleichartige Layouts (aber wiederum gut für Standardnutzer …)
- Mit neuem Mobile-Builder-Tool:

  jquerymobile.com/download-builder/

=> weitere Tools im Web !

Unterstützende Frameworks: Bootstrap

=> getbootstrap.com/

- entstand als internes Projekt von Twitter (Zitat: *„From whiteboarding ideas to coding rough prototypes, collaborating across disciplines is what made Bootstrap successful for internal use at Twitter.“*)
- nach sehr guten Erfolg freigegeben ab 2011 als Opensource Code (MIT-License bis V. 2.x, Apache ab 3.x)

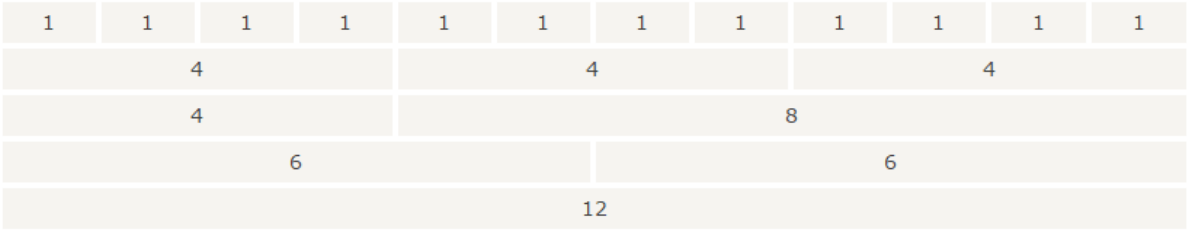
Grundeigenschaften

- Download über obigen Link (Zip)
- modular aufgebaut aus CSS / Javascript / Font-Definitionen
- Bootstrap ist „mobile-first: the code for Bootstrap now starts by targeting smaller screens like mobile devices and tablets, and then "expands“ components and grids for larger screens.“

```
bootstrap/
├── css/
│   ├── bootstrap.css
│   ├── bootstrap.min.css
│   ├── bootstrap-theme.css
│   └── bootstrap-theme.min.css
├── js/
│   ├── bootstrap.js
│   └── bootstrap.min.js
└── fonts/
    ├── glyphicons-halflings- …
```

Unterstützende Frameworks: Bootstrap

Grundkonzepte



- standardmäßig wird ein 940 Pixel breites zwölfspaltiges Grid-Layout generiert
- Das Grund-Gridelement ist an die Abmessungen der Geräteklassen angepasst
- über die Konfigurationsskripte können alle Abmessungen und Rahmen und angepasst werden
- Vordefinierte Größenkennungen :
  - **xs (extra small -> phones <768px)**
  - **sm (small -> tablets (>=768px))**
  - **md (medium -> desktops (>=992px))**
  - **lg (larger -> desktops (>=1200px))**
- Anwendung als **Spalten – Klassenname** mit Angabe der Gridspaltenanzahl (die Summe in einer Zeile muss 12 sein !)

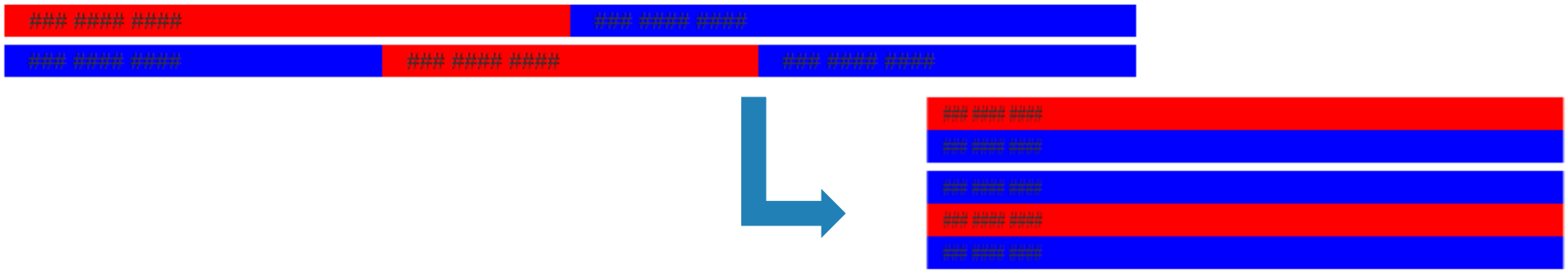
class="col-sm-4“

class="col-md-6"

Unterstützende Frameworks: Bootstrap – Beispiel

Grundkonzept: In einem Container werden Reihen und dann Spalten geschachtelt:

```
<div class="container">
 <div class="row" style="background-color:blue;margin:5px">
  <div class="col-sm-6" style="background-color:red;">### #### ####</div>
  <div class="col-sm-6" style="background-color:blue;">### #### ####</div>
 </div>
 <div class="row" style="background-color:red;margin:5px">
  <div class="col-sm-4" style="background-color:blue";>### #### ####</div>
  <div class="col-sm-4" style="background-color:red;" >### #### ####</div>
  <div class="col-sm-4" style="background-color:blue;">### #### ####</div>
 </div>
```



Unterstützende Frameworks: Bootstrap – Tools

Bootstrap offers also collections of predefined objects and tools

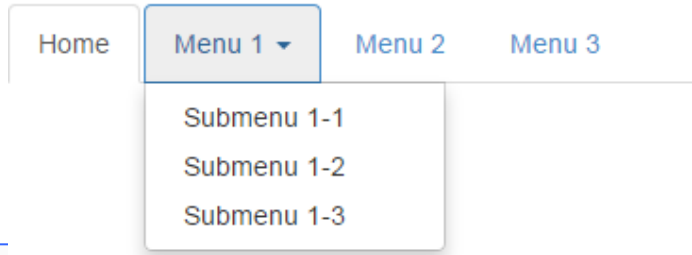
- **Buttons**

  - with classnames: btn-default btn-primary btn-success btn-info btn-warning btn-danger btn-link
  - Usage: <button type="button" class="btn btn-success">Success</button>
  - with buttons-sizes : btn-lg btn-md btn-sm btn-xs
  - Weitere Optionen: Button-groups, deactivated-buttons

- **Glyphicons (spezielle Icon aus der glyphicons.com –Lib als free Versionen)**
  -
  - auch in Kombination mit Buttons
     Print

- **Navigations-Menüs (auch mit Submenüs)**

```
<ul class="nav nav-tabs"><li class="active"><a href="#">Home</a></li>
 <li class="dropdown"><a class="dropdown-toggle" data-toggle="dropdown" href="#">Menu 1
 <span class="caret"></span></a> <ul class="dropdown-menu">
        <li><a href="#">Submenu 1-1</a></li> </ul>  …
    </li>  <li><a href="#">Menu 2</a></li>
  <li><a href="#">Menu 3</a></li>  </ul>
```



Unterstützende Frameworks: Bootstrap – Images / Forms

- **Images** with shaped corners

```
<img src="12-0230.jpg" class="img-rounded" >
<img src="12-0230.jpg" class="img-circle" >
<img src="12-0230.jpg" class="img-thumbnail" >
```

- **Forms**

Email: Enter email Password: Enter password Remember me Submit

  - with some global styling (rounded corners / with placeholders)
  - layout options : vertical (default) / inline / horizontal (like in example)
  - support for all HTML5 input types: text, password, datetime, datetime-local, date, month, time, week, number, email, url, search, tel, and color.

```
<div class="form-group"><label for="usr">Name:</label>
<input type="text" class="form-control" id="usr"></div>
<div class="form-group"><label for="pwd">Password:</label>
<input type="password" class="form-control" id="pwd"></div>
```

Name: test

Password: ••••••••••

  - with additional feedback information

```
<div class="form-group has-warning has-feedback">
    <label for="inputWarning2">Input with warning</label>
    <input type="text" class="form-control" id="inputWarning2">
    <span class="glyphicon glyphicon-warning-sign form-control-feedback"></span>  </div>
```

  - There is a lot of additonal paramters for with additional feedback information

Bootstrap – Less support

The generation of complex CSS-styles is supported by a CSS- preprocessor (like a CSS-compiler), or integrated as a JS-package at the client or at the server-side (e.g. some comand line tools) - see http://lesscss.org/ for details

```
@myColor: #143352;

@the-border: 1px;

 #header {   background-color: @myColor;  }
h2 {      color: @meineFarbe;
          border-left: @the-border;
          border-right: (@the-border * 2); // also calculations !!!
 }
```

There are more options for mixing styles or defining styles in function-like manner.

Hardwarespezifische Anwendungsanforderungen

Energie-spezifische Anforderungen

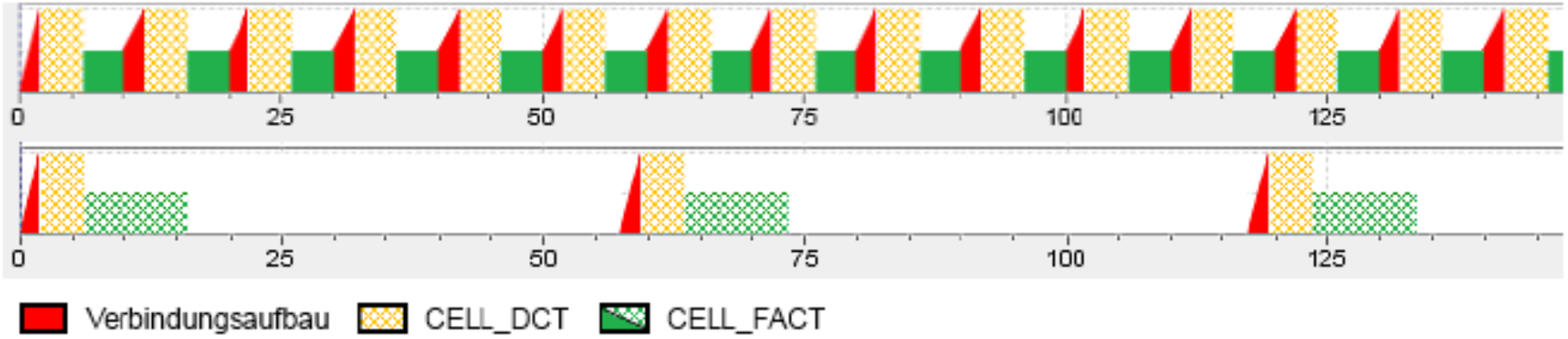
- Hauptproblem ist die begrenzte **Akku-Kapazität** (User-Eindruck!!!!)

Das Netzwerkmodem arbeitet in verschiedenen Stati:

- VOLL-AKTIV (CELL_DCH) - volle Leistung
- Low-Level (CELL-FACH) – mit geringer Leistung (langsamer)
- Leerlauf (IDLE) - sehr sparsam
- **Die Umschaltung zw. den Stati dauert meist einige Sekunden !**

Folge und Anforderungen:

- Keine Pollingaktionen im Sekundenbereich , da sonst kein IDLE-Status!



Polling-Alternativen (speziell für DB-Apps)

- **Long-Polling:** es wird ein spezieller (XHR-) Request abgesetzt, auf welchen der Server nicht sofort antwortet, erst bei neuen Daten oder drohendem Timeout (30 … 60s) kommt eine Antwort -> nicht sehr viel besser und Detailprobleme (Netzwerkausfall/Proxies/ TO´s)

- **Server-Sent-Events**: durch Setzen eines speziellen Flags im HTTP-Header MIME: text/event-stream signalisiert der Server das zukünftige Senden von Events
  - Die Event-Nachrichten werden als Textblöcke versendet:

    Id: ID1

    Data: Nachrichtentext (UTF8-kodiert), ggf. weitere Nachrichtenzeilen

    Retry: 5000 - erneutes Event nach 5 s (oder anderer Wert)
  - **Bis zu 30 Minuten lang offener Kanal !**

Existierende, spezielle Anwendungsprotokolle : XMPP

- **XMPP - Extensible Messaging and Presence Protocol**
  - startete 1999 mit der Entwicklung von **Jabber** (offenes Instant Messaging)
  - Protokoll für echtzeitnahes Extensible-Markup-Language-(XML)-Streaming
  - ab 2005 Einsatz durch Google Inc. in Google Talk (VoIP / IM)
  - es wird generell ein Server benötigt (vers. Versionen als freie Software)
  - zwischen Client und Server wird ein XML-Stream aufgebaut
    - Über diesen werden kleinere XML-Parts u.a. Presence und Statusinfo und die eigentlichen Nachrichten (**XML-Stanzas** vgl. http://xmpp.org/rfcs/rfc3921.html) ausgetauscht, :

Verbindungsanfrage an Server

```
<iq to='example.com' type='set'
   id='sess_1' >
<session xmlns='urn:ietf:params:
xml:ns:xmpp-session'/> </iq>
```

Messageaustausch

```
<message to='romeo@example.net'
from='juliet@example.com' type='chat'
> <body>Where are you, Romeo?
</body> </message>
```

Verbindungsbestätigung

```
<iq from='example.com'
type='result'
id='sess_1'/>
```

Präzenzinfos (Status)

```
<presence>
<show>away
</show> </presence>
```

Statusoptionen

- chat (bereit zum Chat)
- away
- dnd ("Do Not Disturb")

Existierende, spezielle Anwendungsprotokolle

- **MQTT – Message Queuing Telemetry Transport**
  - startete ebenfalls 1999 bei IBM und Arcom zur Überwachung von Ölpipelines
  - heute auch Unterstützung von Sensornetzwerken, Maschine-zu-Maschine-Kommunikation und der Einsatz in der Telemedizin
  - Kommunikation über ein Kontext Broker mit einem ereignisgesteuerten Protokoll (MQTT Protokoll v3.1.1 is seit 2014 OASIS Standard)
  - Broker verbindet verschiedene Gerätetypen (z.B. Sensor / Smartphone)
  - wahrscheinlich eingesetzt beim Facebook Messenger
  - Hierarchische strukt. Messages: **sensors/COMPUTER/temp/HARDDRIVE_1**
  - besonders geeignete für kleine Sensornetzwerke (kein XML-Overhead)

- **AMQP - Advanced Message Queuing Protocol**
  - startete ca. 2003 im Bankberech (bei JP Morgan Chase Bank) zum Hochfrequenz-datenaustaqsuch (im Börsenbroker-Bereich)
  - besonders Augenmerk auf Performance, ebenfalls OASIS-Standard

Universelles Kommunikationsprotokoll - WebSockets

- **WebSocket** ist ein neues Protokoll der Anwendungsschicht aus dem HTML5 - Bereich (erste Standards von 2011 W3C)
- realisiert eine **Voll-Duplex-Verbindung** über Sockets (TCP-Verbindung in beide Richtungen zw. Client und Server
- beim Verbindungsaufbau erfolgt eine clientseitige HTTP GET-Anfrage, welche die Erweiterung „Upgrade auf WebSocket“ enthält:
- Danach können relativ beliebige Textnachrichten gesendet werden
- Verbindungen können auch auf Port 80 laufen (reduziert Probleme mit Firewalls etc.)

Verbindungsanfrage an Server

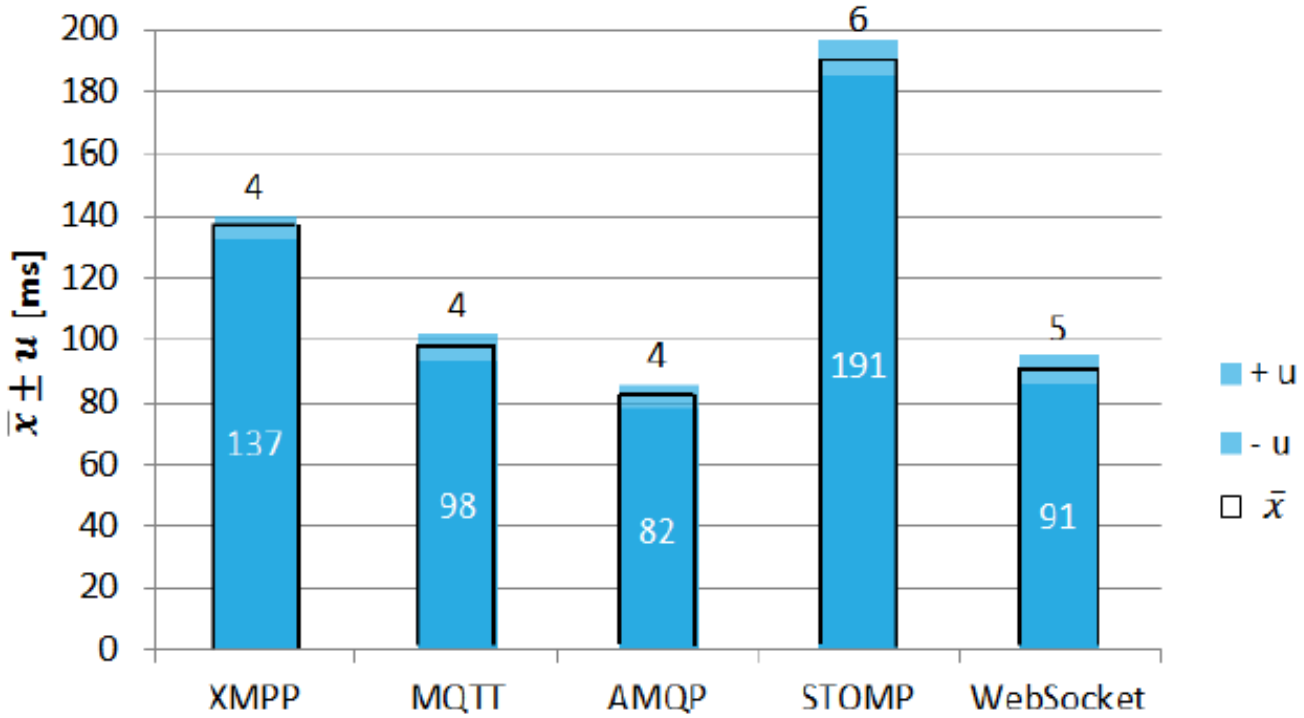
```
GET /chat HTTP/1.1
Host server.example.com
Upgrade: websocket
Connection: Upgrade
```

- Auf Client und Server muss jedoch ein eigenes Protokoll (Generierung der Nachrichten und Parsen) erstellt werden

Performancevergleich der Kommunikationssprotokolle

- **Die diskutierten Protokoll wurden bzgl. Ihrer Performance untersucht** **(vgl. Diplomarbeit** Victoria Fels "Markt- und Produktanalyse verfügbarer Server-Push-Protokolle zur echtzeitnahen Übertragung von Nachrichten auf mobile Endgeräte")

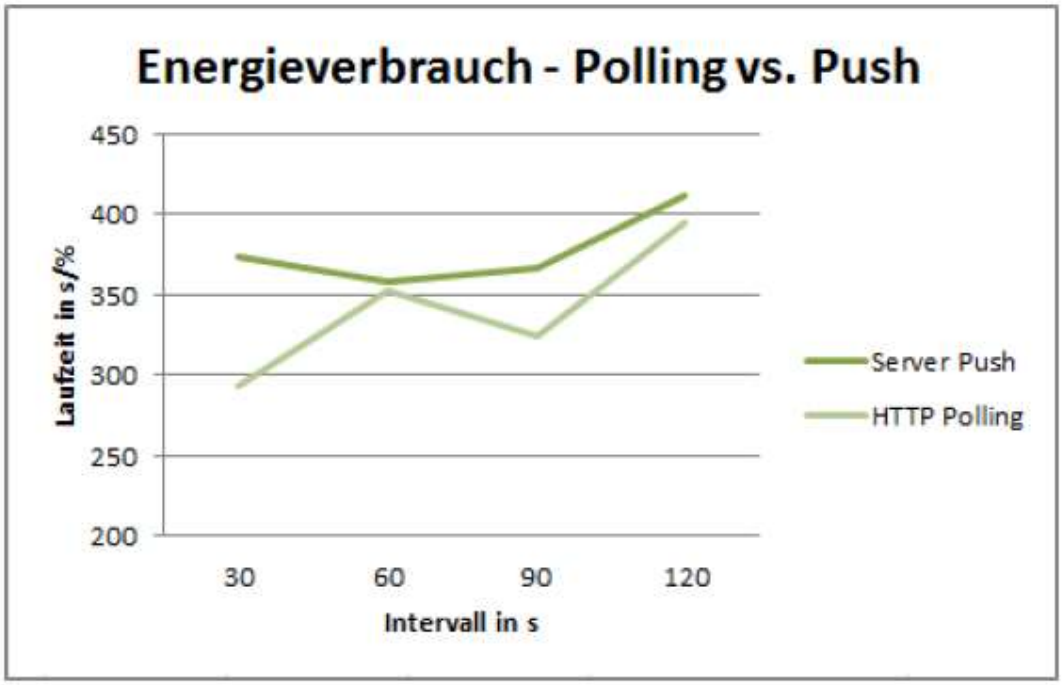
Vergleich der Roundtrip-Zeiten



Energievergleich der Kommunikationssprotokolle

- **Die diskutierten Protokoll wurden bzgl. Ihres Energieverbrauchs untersucht** **(vgl. Diplomarbeit** Anna Pieper "Detailevaluation verfügbarer Server-Push-Protokolle zur echtzeitnahen Übertragung von Nachrichten auf mobile Endgeräte")

Vergleich der Zeiten bis zu einer Reduzierung des Akkus um 25%



Zukünftige Entwicklungen - WebRTC / Http2

- **WebRTC (Web Real-Time Communication)**
  - **wird seit 2011 entwickelt (unter Ägide des W3C)**
  - **speziell auch zum effizienten Austausch von Echtzeitmultimedia**
  - **getUserMedia - > capture media**
  - **RTCPeerConnection / RTCDataChannels zum Echtzeit-Datenaustausch auf über Peer-to-Peer-Netzwerke (->Video Chat etc. ?)**
- **Seit 2015 existiert http 2.0**
  - **darin sind auch neue Protokolle für Server Push enthalten**
  - **vgl. https://http2.github.io/http2-spec/#StreamsLayer**
  - **mit weiteren neuen Befehlen zur Flow-Kontrolle der Streams**
  - **Möglichweise werden auch schon bestehende Protokolle (Websockets, WebRTC) einfliessen**

➔ **diese Technologien sollten die nächsten Jahre genau beobachtet werden**

Auswirkungen auf das Projektmanagement

-> Neue Managementanforderung nach Umfrage

http://www.creativebloq.com/responsive-web-design/problems-8122790

Hauptprobleme aus Sicht der Entwickler:

1. Explaining RWD to clients
2. The lack of a static design phase
3. Navigation
4. Images
5. Tables
6. Converting old fixed-width sites
7. What to serve users of old versions of IE
8. Testing time and cost

Zusammenfassung

Mobile Webapp- Entwicklung

- **Noch stark im Prozess der Entwicklung und Anpassung**
- **mit Responsive UND Adaptive Design existieren 2 schon recht gute Methoden**
- **zukünftige Geräte (auch aus dem VR-Umfeld) können weitere Anpassungen und völlig neue Ansätze erfordern**
- **Sinnvolle Tools zum Prüfen sind notwendig**
  - **Mobile-Web-Validatoren (analog W3C-Validator)**

  => *auch Abschlussarbeiten im gesamten mobilen Bereich !!!*

=> VIEL SPASS und ERFOLG – es bleibt weiterhin spannend …