BERUFSAKADEMIE SACHSEN
STAATLICHE STUDIENAKADEMIE
DRESDEN
UNIVERSITY OF COOPERATIVE EDUCATION

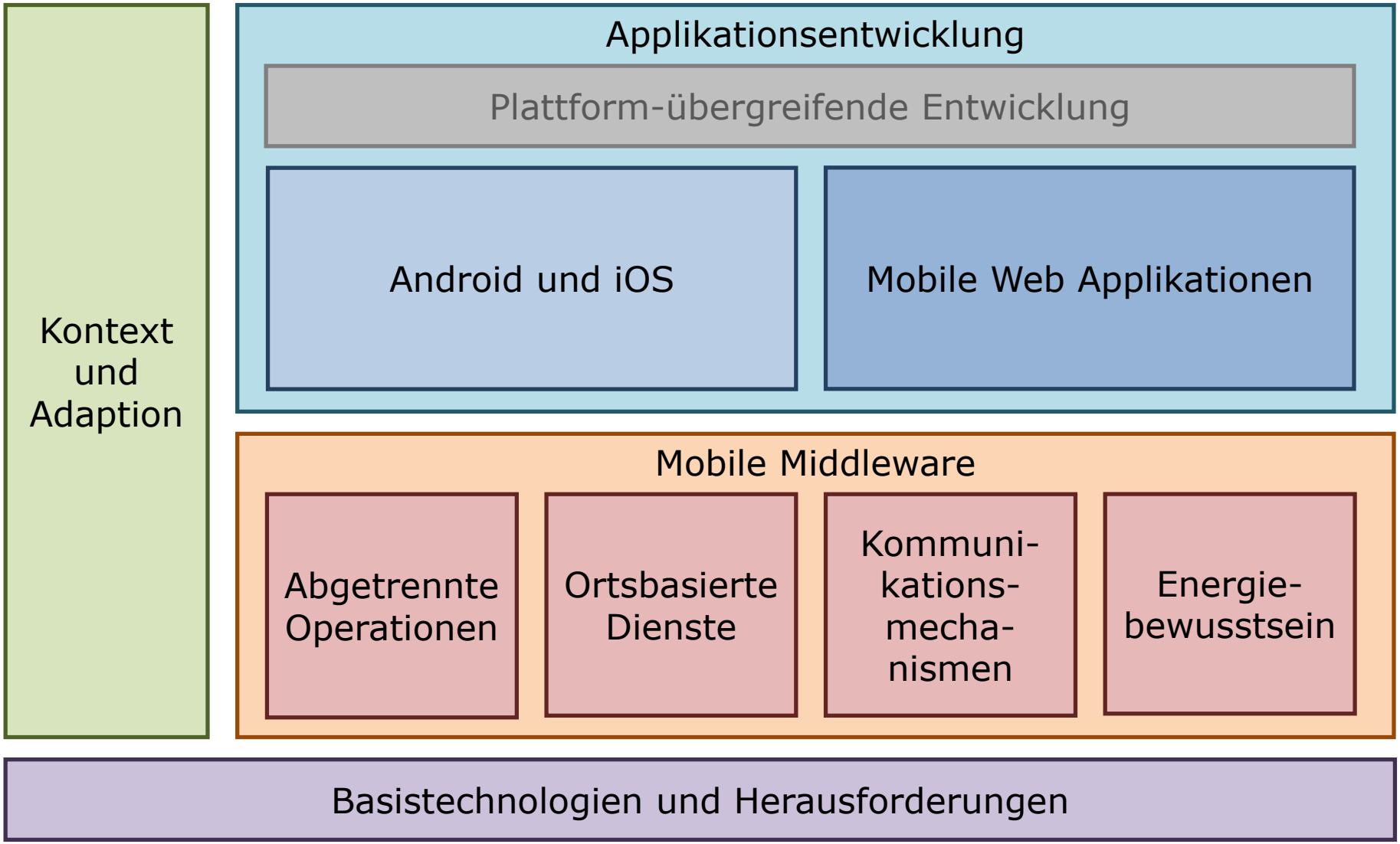
Web- und App-Programmierung

Mobile Web-Applikationen

mit Skriptmaterial von Dr.-Ing. T. Springer

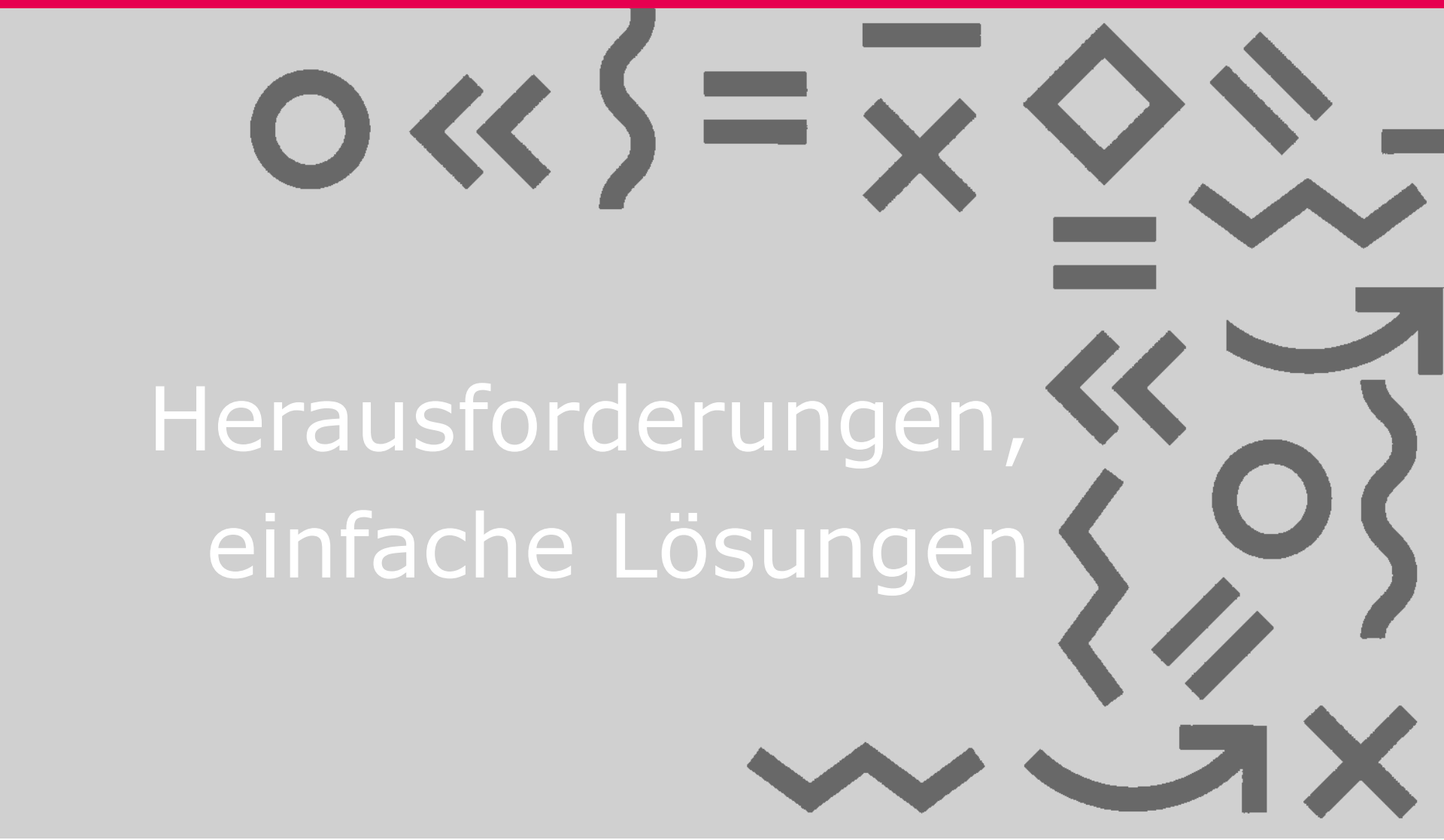
Prof. Dr.-Ing. Tenshi Hara
tenshi.hara@ba-sachsen.de

Aufbau der Lehrveranstaltung



Übersicht

- Motivation und Herausforderungen
- Hauptprinzipien
  - Responsive Web Design
  - Mobile First
  - Design-Muster
- weitere Eigenschaften in HTML 5
- serverseitige Adaptation



Herausforderungen, einfache Lösungen

Herausforderungen des Web-Designs für mobile Geräte

- Abhängigkeit von Netzwerkverbindung
- Unterstützung heterogener Endgeräte (Desktop, Tablet, Smartphone)
  - unterschiedliche Bildschirmgrößen
    (bspw. 2023: 27% HD (davon 9% FullHD), 39% SD, 43% Sonstige)
  - natives Aussehen und intuitive Bedienung
  - Berührungs-/Gesten-Eingaben
  - Zugriff auf native Funktionalität (Lokation, Kamera, …)
- klassischer Ansatz: unterschiedliche Versionen für unterschiedliche Endgeräte (mobile.example.com, app.example.com, www.example.com)

Einfache Lösungen

Die einfachsten Lösungen liegen in den Web-Technologien.

→ 3MI-GlWeb-20 ←

Responsive Web Design

- Vermeiden separater Mobil-Version → eine Web-Seite bedient alles
- Anpassung an Bildschirmgröße via CSS Media Queries
- Hauptkonzepte
  - Fluid Grids
  - Flexible Images
  - CSS Media Queries

http://en.wikipedia.org/wiki/File:Boston_Globe_responsive_website.jpg

Mobile First (1/3)

das mobile Web ist der Markt der Zukunft:
"Mobile phones will overtake PCs as the most common Web access devices worldwide by 2013" [Gartner]

→ im Dezember 2016 tatsächlich geschehen (siehe Kapitel 0)

Konsequenz

- Optimierung der Inhalte für mobile Endgeräte
- fortgesetzte Unterstützung von Desktop-Rechnern
- Fokus auf wichtige Inhalte und ihre Platzierung
- Abstimmung auf begrenzte Ressourcen

→ mobile Optimierung kommt auch Desktops zu Gute

Mobile First (2/3)

Grundprinzip: Entwicklung beginnt mit Ziel mobiler Unterstützung

- (physisch) kleiner Bildschirm
  - kondensierte Navigation und Inhalte (Reduktion auf die wichtigsten Aspekte)
  - responsives Layout (Responsiveness)
- langsames Netzwerk
  - Vermeidung von Anfragen und verringerte Übertragungsmenge
  - Kombinationsbilder (Image Sprites), einzelne CSS/JS-Dateien, minimierte CSS/JS-Dateien (Minified Source)
  - lokales Cachen von Informationen (ApplicationCache, localStorage)
  - CSS 3, <canvas> oder SVG statt rasterbasierter Bilder
- mobile Gerätenutzung
  - Optimierung für möglichst einhändige Nutzung
  - partielle Nutzeraufmerksamkeit

Mobile First (3/3)

- Berücksichtigen der Gesteneingabe von Anfang an
  - Zielgröße für Gesten (Touch Target Size)
    - Empfehlung: min(9mm, 34px)
    - Minimum: min(7mm, 26px)
  - visuelle Größe ist 60% bis 100% der Zielgröße
  - Unterstützung von Mehrfingergesten (Multi Touch)
    → `Tap|Double Tap|Drag|Flick|Pinch|Spread|Rotate`
  - Schnittstellenparadigma: durch Wischbewegungen offenlegen, neu laden, … (Drag to Reveal, Drag to Refresh, ...)
- Geräteeigenschaften
  - Lokation: Anpassen der Inhalte an Nutzerlokation
  - Kompass: Augmented Reality anhand der Lokation und Richtung der Nutzenden
  - Ausrichtung (Hoch- und Querformat): Optimierung des Layouts (Navigation, Inhalt) für die Geräteausrichtung
  - Kamera: Nutzung von Bilder für Eingaben (bspw. QR-Code)

Technologien von AJAX

AJAX ist keine einzelne Technologie, sondern Kombination von mehreren (entsprechend Definition von J. J. Garret)

- standardisierte Präsentation mittels (X)HTML und CSS
- dynamische Anzeige und Interaktion durch DOM-Manipulation
- Datenaustausch und -manipulation mittels XML und XSLT
- asynchrones Laden von Daten mittels `XMLHttpRequest`-Objekt
- JavaScript führt und bindet alles zusammen

⇒ übertragbar auf andere Austauschformate, bspw. JSON statt XML

⇒ übertragbar auf andere Transferprotokolle, bspw. XML via WebSocket anstelle von XMLHttpRequest

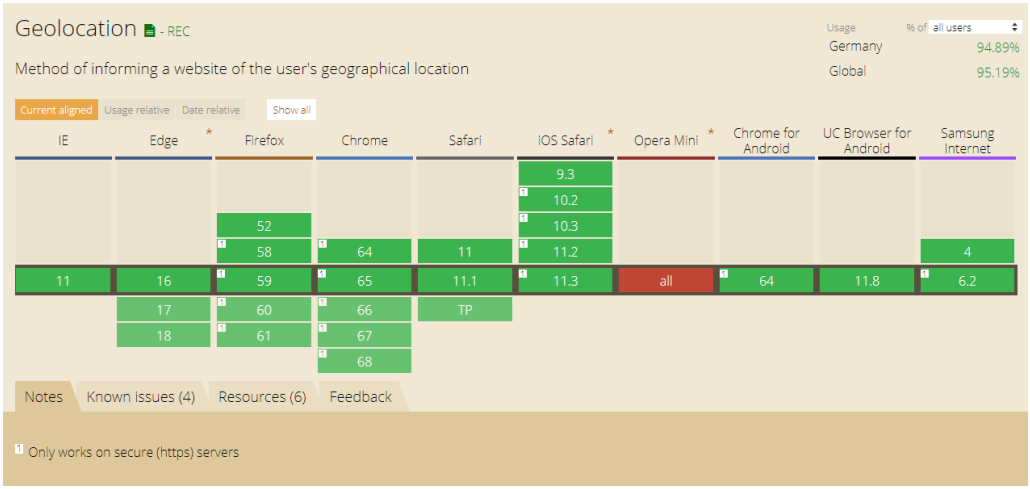
Funktionalität der AJAX-Engine

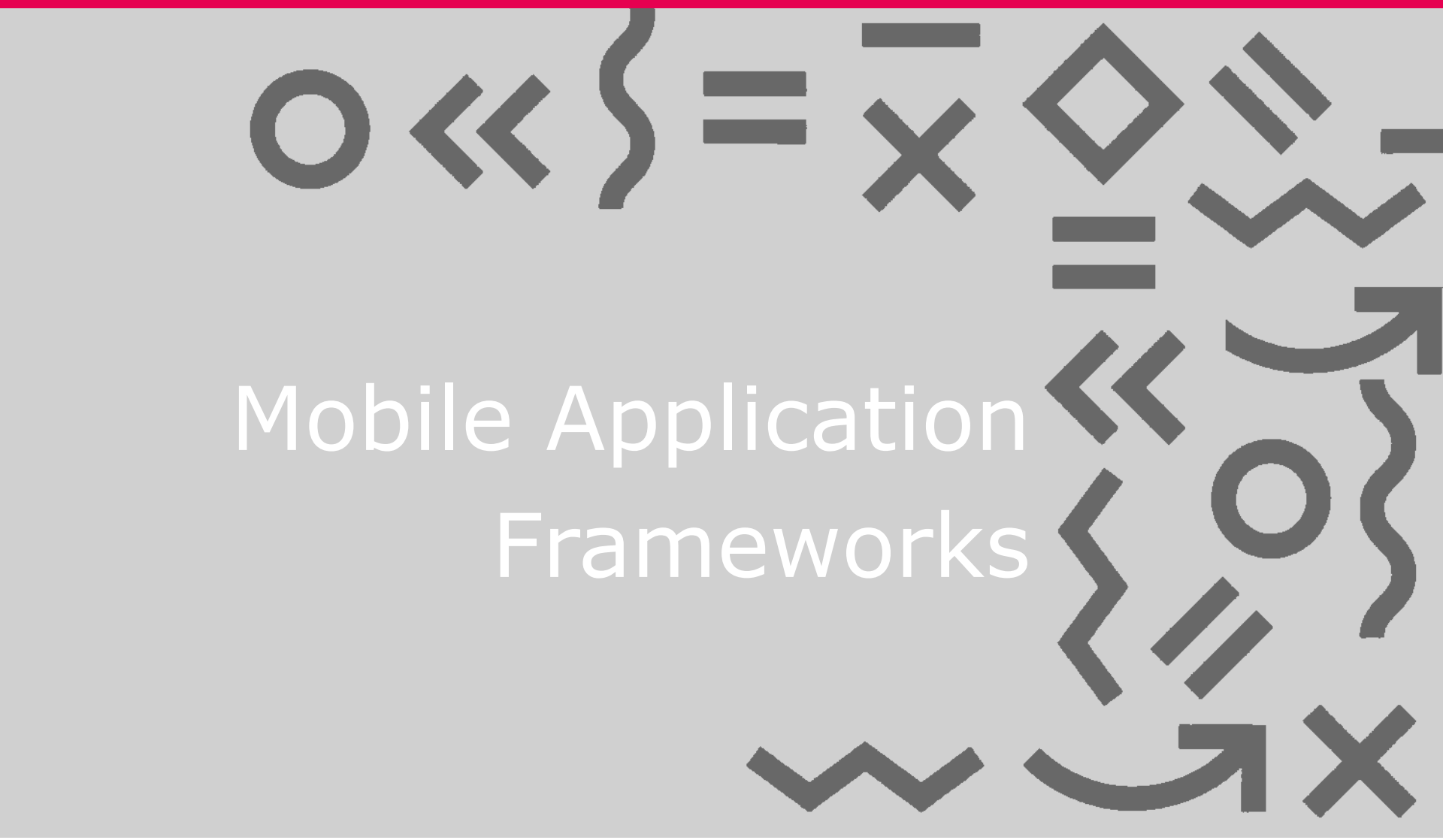
- Nutzeraktionen generieren JavaScript-Aufrufe zur AJAX-Engine anstelle von HTTP-Anfragen an den WWW-Server
- nicht jede Antwort auf Nutzeraktionen muss vom WWW-Server kommen; AJAX-Engine operiert selbständig und kann auf lokalen Daten arbeiten
- Server-Anfragen reduzieren sich auf Fälle in denen AJAX-Engine
  - etwas vom Server benötigt um Nutzeraktionen zu bearbeiten (d.h. lokal unzureichende Daten oder fehlende Prozedur)
  - Datensynchronisierung mit Server für notwendig hält oder erzwingt (bspw. Ergebnis multipler Aktionen benötigt Server-Bestätigung)

⇒ AJAX-Engine macht alle Anfragen asynchron

HTML 5 – aktueller Stand

- seit 28. Oktober 2014 Standard des W3C
- im Sprachgebrauch ist „HTML 5" mehr als HTML5 (Canvas, WebGL, …)
  - viele beteiligte Technologien sind (noch) nicht standardisiert
  - Nutzbarkeit von „HTML 5" hängt von zwei Faktoren ab
    - Implementierung im Browser
    - Standardisierung von Schnittstellen zwischen den Implementierungen
- Fähigkeiten des Browsers testen: www.caniuse.com





Mobile Application Frameworks

Entwicklung mobiler Web-Applikationen

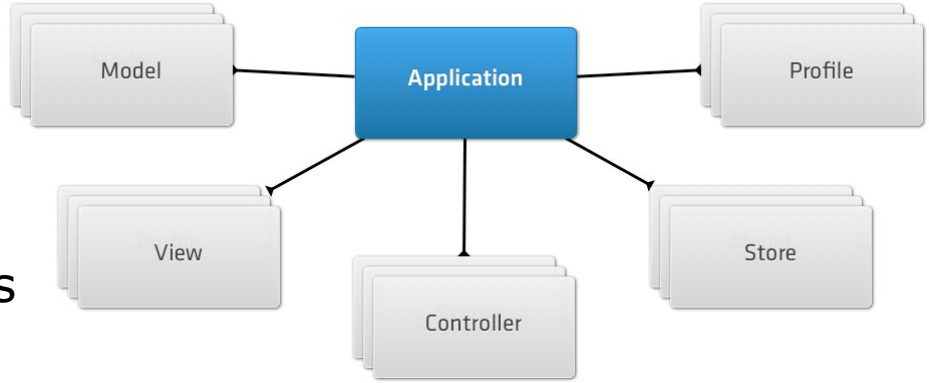
- Responsive CSS
  - Bootstrap [twitter.github.com/bootstrap]
  - jQuery UI [jqueryui.com]
  - Foundation [foundation.zurb.com]
  - u.v.m.
- Mobile Web Application Frameworks
  - Sencha Touch [www.sencha.com/products/touch]
  - jQuery Mobile [jquerymobile.com]
  - M-Project [the-m-project.org]
  - LungoJS [www.lungojs.com]
  - Joshfire [framework.joshfire.com]
  - u.v.m.
- Testen und Debuggen
  - Ripple [ripple.tinyhippos.com]
  - Adobe Edge Inspect [html.adobe.com/edge/inspect]
  - u.v.m.

⇒ diverse plattformintegrierte und plattformübergreifende Lösungen

⇒ teilweise im Browser integrierte Tools zum Entwickeln und Testen

Sencha Touch

- MVC-JavaScript-Framework für mobile Applikationen
- nativ aussehende HTML-5-Applikationen
- unterstützt Paketierung als native iOS/Android-App durch PhoneGap
- unterstützte Plattformen: iOS, Android, BlackBerry, Windows Phone
- bietet auf ExtJS basierendes Klassensystem
  → pures JavaScript-Framework für plattformübergreifende Applikationen
- Applikationen bestehen aus
  - Models
  - Views
  - Controllern
  - Stores (Speicher für Modell-persistierung und clientseitiges Cachen von entfernten Daten)
  - Profiles/Themes (Geräteklassen für GUI)



Sencha Touch – Konzepte

- Component (GUI-Klasse)
  - Schnittstellen-Widgets (Unterklasse von `Ext.Component`)
  - in Containern geschachtelt oder schwebend (float)
  - bspw. Panel, Form, DataView, List, Map, ...
- Layout (Größe und Position von Components)
  - definiert innerhalb von Containern
  - flexibles vertikale/horizontales Raster (Karten-Layout)
  - unterstützt Docking
- Event (Ereignisse)
  - treten bei Änderungen an Components auf
  - anpassbare Trigger/Listener
  - ereignisbasiertes Scoping und Zwischenspeichern
- Erkennung der Laufzeitumgebung
  - Plattform (`Ext.os.name`)
  - Browser (`Ext.browser.name`)
  - Browser-Eigenschaften (Media Queries, Geolocation, Touch, ...)

```
var panel =
Ext.create('Ext.Panel', {
    html: 'Beispiel-Panel'
});
Ext.Viewport.add(panel);
```

Sencha Touch – Touch-Ereignisse

- Unterstützung diverser Gesten
  - durch Event-Listener umgesetzt
  - Listener werden in Setzreihenfolge ausgelöst
- Event-Typen
  - touchstart
  - touchmove
  - touchend
  - dragstart
  - dragend
  - tap
  - singletap
  - doubletap
  - longpress
  - swipe
  - pinch
  - rotate

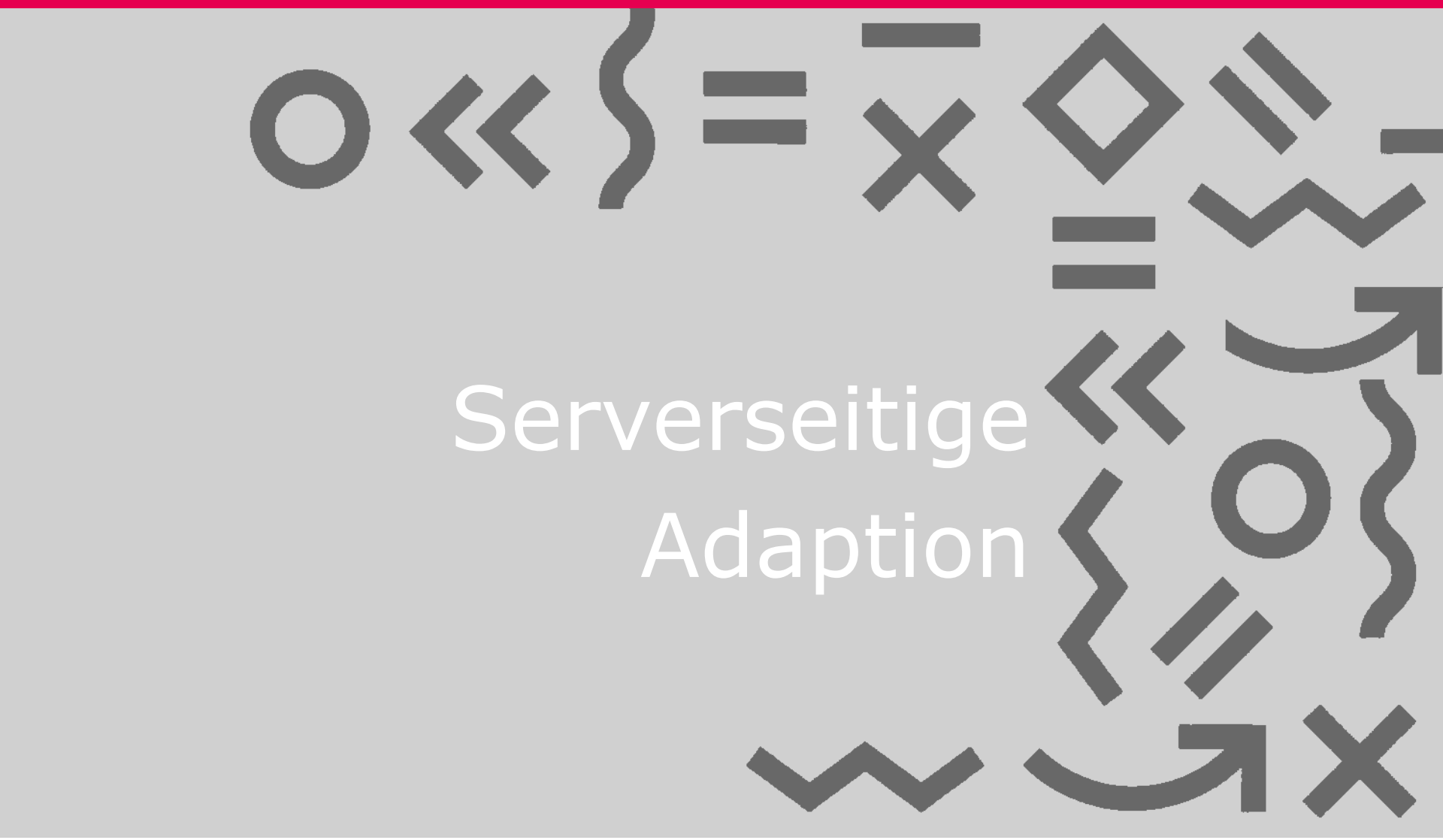
```
// Event-Listener mit .on() setzen
myButton.on("tap", function(event) {
    alert("Schaltfläche wurde getapt.");
});

// mittels Pinch Zoomen

// „pinchstart“ wird einmalig ausgelöst
myElement.on("pinchstart", function(event) {
    this.baseWidth = this.getWidth();
    this.baseHeight = this.getHeight();
});

// solange Gepincht wird, wird „pinch“
// kontinuierlich ausgelöst
myElement.on("pinch", function(event) {
    var scale = event.scale;
    this.setWidth(this.baseWidth * scale);
    this.setHeight(this.baseHeight * scale);
});

// „pinchend“ wir am Ende einmalig ausgelöst
myElement.on("pinchend", function(event) {
    alert("Pinch-to-Zoom abgeschlossen.");
});
```

Sencha Touch – Nutzerschnittstellengestaltung

- Aussehen hängt von CSS-Style und Component-Layout ab
  (bspw. blaue Navigationsleiste unter iOS, schwarze unter Android)
- Styling in CSS-Datei = Theme
- plattformspezifisches Styling durch vordefinierte Themes bereitgestellt
  - sencha-touch.css
  - apple.css
  - android.css
- Component-Layout wird für jede Plattform durch Profiles definiert
  (bspw. Tablet vs. Smartphone, Hoch- vs. Querformat)
  → beim Start der Applikation aktiviert
  (ggf. auch bei Ausrichtungsänderung)

Sencha Touch – Nutzerschnittstellengestaltung

```
// Theme auf Basis der Geräteplattform auswählen
Ext.setup({
    onReady:function( ) {
        if (Ext.is.Android)
            Ext.get("myStylesheet").dom.href = "android.css";
        else if(Ext.is.iOS)
            Ext.get("myStylesheet").dom.href = "apple.css";
    }
});
// Views auf Basis des Profiles ändern
// auf Smartphone ganzen Viewport füllen, auf Tablets Modal fester Größe nutzen
Ext.define("view.phone.Form", {
    extend: "view.Form",
    config: { title: "Version für Smartphones" }
});
Ext.define("view.tablet.Form", {
    extend: "view.Form",
    config: {
        title: "Version für Tablets",
        modal: true,
        height: 1145,
        width: 720,
        centered: true
    }
});
```

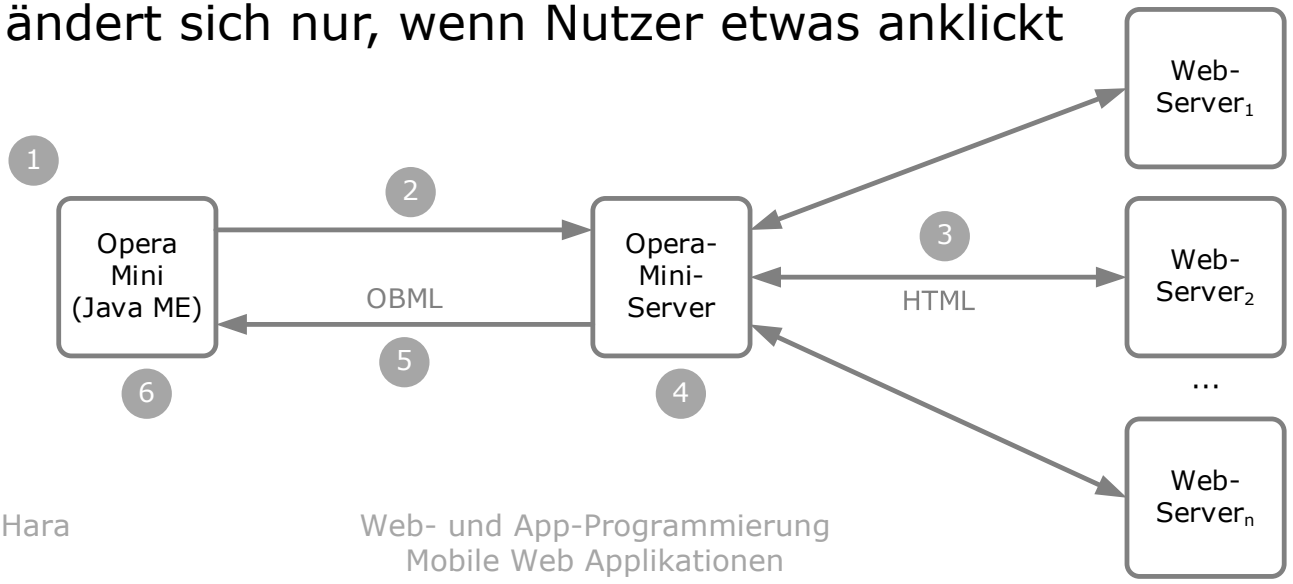


Serverseitige Adaption

serverseitige Adaption

Beispiel: Opera Mini

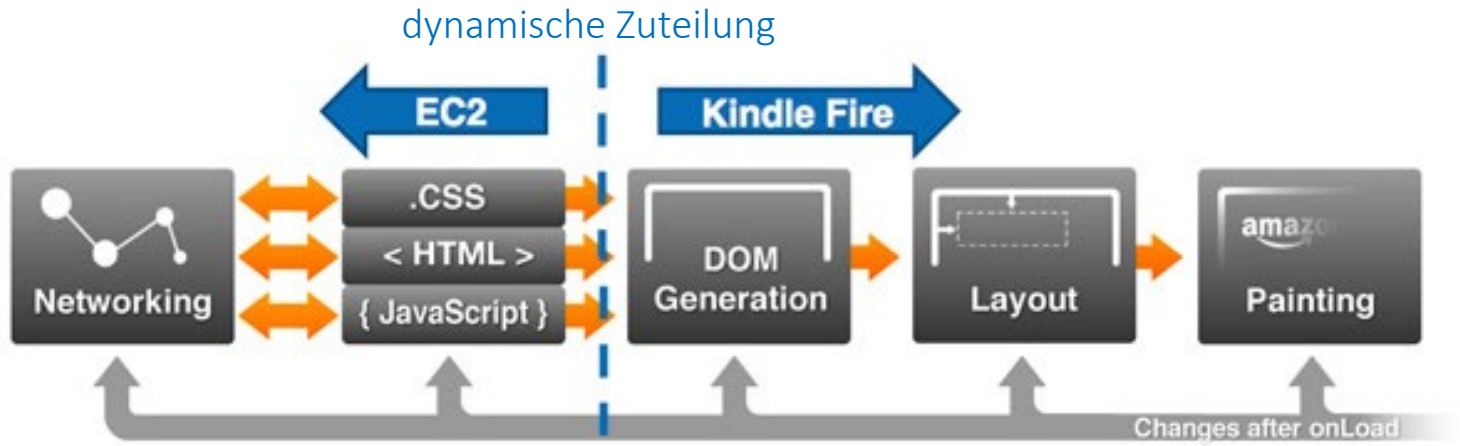
- angeforderte Web-Seite wird auf Proxy-Server verarbeitet
- fertig gerenderte Seite wird in Opera Binary Markup Language (OBML) komprimiert und an Client gesendet
- reduziert zu übertragende Datenmenge um bis zu 90%
- Grenzen des Ansatzes
  - nur onLoad-Ereignisse (serverseitiges JavaScript) voll unterstützt
  - auf Client nur Nutzereingaben (`onClick`, `onChange`, ...) unterstützt
  - begrenzte AJAX-Unterstützung
    → Seite ändert sich nur, wenn Nutzer etwas anklickt



serverseitige Adaption

Amazon Silk

- Browser auf Fire-Tablet
- Browser-Subsysteme (Fetching, JavaScript, Rendering) auch als Proxys in der Elastic Compute Cloud (EC2) implementiert
- dynamische Ausführungszuteilung zu EC2- und Endgerät-Subsystemen
- Vorteile der Cloud
  - berechnungsintensive Aufgaben (Rendering, Skripte) nicht auf Endgerät
  - schnelleres Holen von referenzierten Dateien (CSS, JavaScript, Bilder, ...)
  - gute Vorhersagen durch Aktivitäten aller Silk-Nutzer (Predictive Caching)
  - optimierte Bildkompression (da Silk Browser und Endgerät kennt)



Zusammenfassung und Aufgaben

Zusammenfassung

- adaptive Web-Applikationen für mobile Geräte
- Hauptprinzipien
  - Responsive Web Design
    - skalierbare Layouts und Grafiken
    - alternative Layouts und Inhalte anhand Media Queries
  - Layout-Muster für unterschiedliche mobile Geräte
  - Mobile First → Inhalte für die mobile Nutzung optimieren
- HTML 5 bietet Voraussetzungen für mobile Optimierung
  - Gerätezugriff, CSS3, Multimedia, Offline-Speicher, …
  - erweiterte Browser-Fähigkeiten
- Inhaltsadaption in der Infrastruktur
  - Verringerung des zu übertragenden Datenvolumens
  - Unterstützung serverseitiger Verarbeitung

Aufgaben

- Erstellen Sie mit Ihren Kommilitonen eine Tabelle mit Informationen zu Ihnen bekannten Webseiten und deren Anpassung zwischen Desktop- und mobilen Zugriffen.
- Die Vor- und Nachteile der serverseitigen Adaption beachtend, erstellen Sie mit Ihren Kommilitonen eine Liste von Anwendungsszenarien, die vom Endanwender (überwiegend) positiv empfunden werden.
- Falls Sie bisher wenig Erfahrung in der Web-Entwicklung sammeln konnten, experimentieren Sie mit den häufig verwendeten Layout-Mustern (*Column Drop*, *Mostly Fluid*, *Layout Shifter*, …) unter https://developers.google.com/web/fundamentals/design-and-ux/responsive/patterns

Referenzen

Crowd-Docs of Web Standards: www1.webplatform.org

Ethan Marcotte: Responsive Web Design, A Book Apart, 2011

Luke Wroblewski: Mobile First, A Book Apart, 2011

Luke Wroblewski: Multi-Device Layout Patterns, http://www.lukew.com/ff/entry.asp?1514

Referenzen

HTML 5 Technologiespezifikationen

- HTML 5: http://dev.w3.org/HTML 5/spec/single-page.html
- CSS: http://www.w3.org/Style/CSS/Overview.en.html
- Microdata: http://www.w3.org/TR/HTML 5/microdata.html
- SVG 2: https://svgwg.org/svg2-draft/
- WebGL: https://www.khronos.org/registry/webgl/specs/1.0/
- Device Access: http://www.w3.org/2009/dap/
- WebSockets: http://www.w3.org/TR/websockets/
- Web Workers: http://www.w3.org/TR/workers/
- XMLHttpRequest 2: http://www.w3.org/TR/XMLHttpRequest/
- ApplicationCache: http://www.w3.org/TR/offline-webapps/
- Web Storage: http://www.w3.org/TR/webstorage/
- Indexed Database: http://www.w3.org/TR/IndexedDB/
- File API: http://www.w3.org/TR/FileAPI/