Version 2023a

BERUFSAKADEMIE SACHSEN
STAATLICHE STUDIENAKADEMIE
DRESDEN
UNIVERSITY OF COOPERATIVE EDUCATION

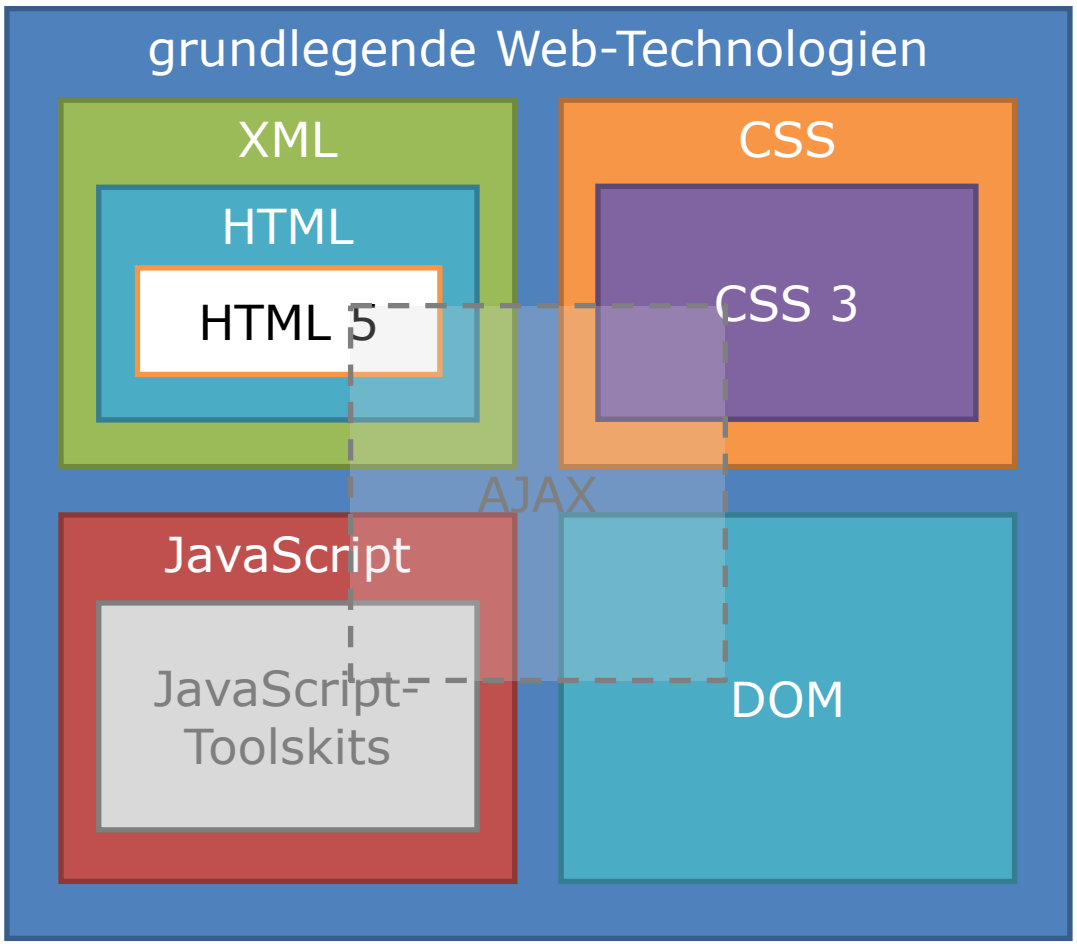
Grundlagen der Web-Programmierung
HTML 5

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

Aufbau der Lehrveranstaltung

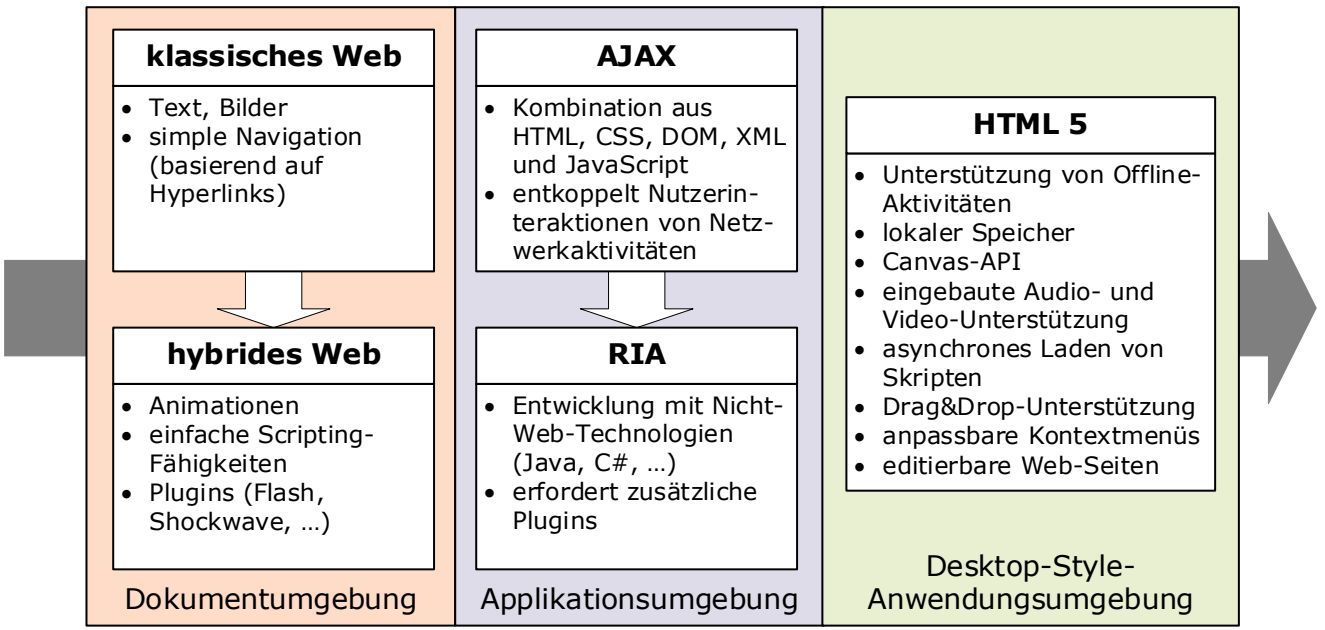
Progressive Web Apps

Search Engine Optimization



Proprietäre Standards

Technologien für Web-Applikationen



HTML5 und CSS 3

- fortgeschrittenes GUI-Styling
- Plugin-freie Multimedia- und Berechnungsfähigkeiten
- Layout-Transformation für multiple Bildschirmgrößen
- Unterstützung von Offline-Aktivitäten
- alternative Mediendateien

HTML 5: zugeschnitten für Mobile-First-Entwicklung

- basierend auf XHTML 1.1 und HTML 4
- verwaltet durch W3C und WHATWG
- neue Elemente und Attribute
- neue Technologien; in Sektionen modularisiert
  → unter HTML 5 beworben; i.d.R. aber eigene Standards


Semantics


Multimedia


Offline & Storage


3D, Graphics & Effects


Device Access


Performance & Integration


Connectivity


CSS 3.0

- mehr Fähigkeiten für Web-Applikationen
- Verringerung des Bedarfs an proprietären Plugins
  → Plugins sind auf mobile Geräten i.d.R. nicht verfügbar

Technologien

Multi-resolution Images

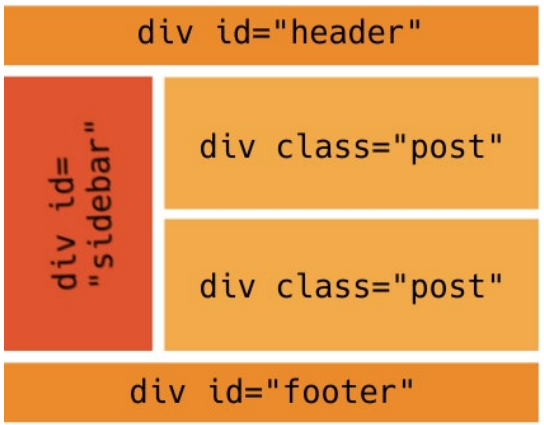
- Problem: Bilder skalieren schlecht über alle Auflösungen (bspw. 151dpi (ZTE Smart Tab 10; 752x1280px) bis 808dpi (Sony Xperia Z5 Premium; 3840x2160px))
  - viele Mobilgeräte benötigen höher auflösende Bilder, als PCs
  - gleichzeitig muss Netzwerkbandbreite berücksichtigt werden
  - Nutzung alternativer Bilder für unterschiedliche Kontexte
- Lösung: `<picture>`-Element
  - multiple Ressourcen an Medienattribute gebunden (z.B. `min-width`)
  - Attribut `src-set` enthält Liste der gebundenen Medien mit Pixel-Verhältnis-Angabe (Verhältnis physischer Pixel zu logischen Pixel)
  - *Beispiel*
    - iPhone: 320 physische Pixel zu 320 logische Pixel (dips) ⇒ 1x
    - retina iPhone: 640 physische Pixel zu 320dips ⇒ 2x

Multi-resolution Images – Beispiel

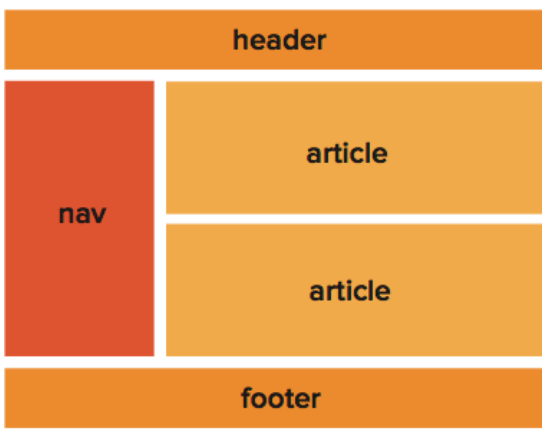
```
<picture alt="Bildbeschreibung">
   <!-- kleines Layout -->
   <source media="(min-width: 18em)"
      srcset="med.jpg 1x, med-highres.jpg 2x" />
   <!-- größeres Layout -->
   <source media="(min-width: 45em)"
      srcset="large.jpg 1x, large-highres.jpg 2x" />
   <!-- Ausweichlösung für ältere Browser -->
   <img src="small.jpg" alt="Bildbeschreibung" />
</picture>
```

SEMANTICS

- semantische Elemente (`<article>`, `<footer>`, `<header>`, `<nav>`, `<section>`) geben dem HMTL-Markup Struktur
- Microdata
  - geben dem HTML-Markup semantische Informationen (Telefonnummer, Adresse, …)
  - `itemscope` definiert Vokabular (schema.org, Rich Snippets, WHATWG, …)
  - `itemprop` weist Elementen Wörter des Vokabulars zu (bspw. `itemprop="name">John Doe`)



HTML 4.01 / XHTML 1.1



(X)HTML5

Multimedia

<video>- und <audio>-Tag zum Abspielen von Medien ohne Plugin

- Attribute: autoplay, controls, muted, …
- multiple Datenquellen möglich (zur Unterstützung alternativer Codecs)
- Codec-Unterstützung variiert zwischen Browsern zum Teil stark
  - Video: theora, h.264, webM, …
  - audio: ogg, wav, mp3, webM, aac, …

```
<video id="meinVideo"><!-- multiple Quellen -->
   <source src="/trailer.mp4" type="video/mp4; codecs='avc1, mp4a'" />
   <source src="/trailer.ogv" type="video/ogg; codecs='theora, vorbis'" />
   Ihr Browser unterst&uuml;tzt kein HTML&bsp;5.<!-- Alternativtext -->
</video>
<button type="button" onclick="vid_play_pause()">&#x25B6;/&#10074;</button>
<script language="JavaScript"><!-- JavaScript-Video-API -->
   function vid_play_pause() {
      var myVideo = document.getElementById("meinVideo");
      if (myVideo.paused) myVideo.play();
      else myVideo.pause();
   }
</script>
```

CSS 3.0

→ siehe Kapitel 3

- fortgeschrittene Gestaltung der (grafischen) Nutzerschnittstelle
- Übergänge und Animationen
- Web Fonts
- Reduktion des Bedarfs an Plugins und Bilder

Device Access

- in früher Entwicklungsphase
- Zugriff auf Sensoren und Speicher der Gerätes
  → Lokation, Kamera, Mikrofon, Bildschirmausrichtung, Bewegungen/Beschleunigungen, Kontakte, Kalender
- Beispiel: Geolocation API [http://caniuse.com/#feat=geolocation]
  - bietet Schnittstelle zu Lokationsinformationen via JavaScript
  - fügt dem DOM `navigator.geolocation`-Element hinzu
  - Nutzer kann den Zugriff explizit erlauben/verbieten
  - in gängigen Browsern unterstützt (IE, Edge, Firefox, Chrome/Chromium, Safari; mobil und Desktop)

Device Access

```
// Prüfen, ob  Geolocation unterstützt wird
if (typeof navigator.geolocation === 'undefined')
   alert("Ihr Browser unterstützt Geolocation nicht.");
else geo = navigator.geolocation;

// Position holen
if (geo) geo.getCurrentPosition(successFunction, errorFunction);

// Lesen des Positions-Objektes und Darstellen als (Lat;Lon)
function successFunction(position) {
   var lat = position.coords.latitude;
   var long = position.coords.longitude;
   var acc = position.coords.accuracy;
   var speed = position.coords.speed;
   var heading = position.coords.heading;
   alert("Ihre aktuelle Position ist: (" + lat + ";" + long + ")");
}

function errorFunction(position)
   alert("Es ist ein Fehler aufgetreten.");
```

CONNECTIVITY

WebSockets (IETF RFC 6455)

- bidirektionaler Kommunikationskanal via TCP
- WebSocket API + WebSocket
- langlebige offene Verbindung ermöglicht Server-Push anstatt Polling
- `HTTP Upgrade` öffnet den WebSocket
- API (JavaScript)
  - Erzeugung eines WebSocket: `new WebSocket(url, protocols)`
  - Methoden: `send(data)` und `close()`
  - Ereignisse: `onopen`, `onclose`, `onmessage`, `onerror`

Performance & Integration

- Web Workers
  - bieten Multi-threading für JavaScript
  - für lange laufende, berechnungsaufwändige, clientseitige Aufgaben
  - unabhängig von Aufgaben der Nutzerschnittstelle
  - kein DOM-Zugriff → Nutzung von Nachrichten (Message Passing)
  - zwei Typen
    - dedicated: an Erzeugerskript gebunden
    - shared: nutzbar für Skripte selben Ursprungs (Same Origin Policy)
- XMLHTTPRequest 2
  - im Vergleich zu v1 (→Kapitel 6) mehr Funktionalität
  - Verfolgung des Übertragungsfortschritts durch Ereignisse
  - binäre Daten-Uploads (in XHRv1 nicht direkt möglich)

Offline & Storage

- Service Worker
  - vom Browser im Hintergrund ausgeführtes Skript; unabhängig von der Web-Seite
  - kann Netzwerkanfragen abfangen und behandeln
  - Zugriff auf lokalen Cache und persistenten Speicher
  - unterstützt Offline-Eigenschaften
- ApplicationCache
  - kontrolliert das Caching durch den Browser
  - konfiguriert via Cache Manifest
  - unterteilt Dateien in `CACHE`, `NETWORK` (nur online) und `FALLBACK`
  - JavaScript-API zur Erzwingung von Cache-Aktualisierungen

Offline & Storage (1/3)

Web Storage

- speichert Schlüssel/Wert-Paare lokal
- JavaScript-API
- Werte können beliebigem von JavaScript unterstützen Typs sein
- Speichergröße beschränkt (abh. vom Browser oder Konfiguration)
- zwei Typen

`localStorage`	kein Ablaufdatum für gespeicherte Daten
`sessionStorage`	Daten werden nach Sitzungsende verworfen

Offline & Storage (2/3)

File API

- Zugriff auf lokale Dateien JavaScript
- Nutzer wählt Dateien durch `<input>`-Element oder Drag&Drop
- einzelne/mehrere Dateien, ganze Verzeichnisse oder binäre Daten (Blob)
- nutzt `FileReader`-Schnittstelle
- referenziert lokale Dateien durch Blob-URIs

Offline & Storage (3/3)

IndexedDB

- Low-Level-API für clientseitigen transaktionalen noSQL-Speicher
- persistente (komplexe) JavaScript-Objekte als Schlüssel/Wert-Paare
- Indexierung von Wert-Eigenschaften
- keine harte Größenbegrenzung (benötigt ggf. Nutzererlaubnis)
- asynchrone JavaScript-API
  - transaktionale Anfragen
  - Auslieferung von Ergebnissen via Rückruffunktionen
  - synchrone API für Web Workers geplant

Local Storage statt Cookies für lokale Daten

- mittels JavaScript adressierbar: `window.localStorage.*`

  `setItem()` - (K,V)-Paar dem Local Storage hinzufügen

  `getItem()` - Wert anhand Schlüssel aus Local Storage lesen

  `removeItem()` - Element mit Schlüssel aus Local Storage löschen

  `clear()` - Local Storage leeren

  `key()` - holt Wert an nummerischem Index aus Local Storage (sehr praktisch für Iterator-Schleifen)

- Prüfen der Local-Storage-Verfügbarkeit mittels

```
if (typeof(Storage) !== "undefined") {
   // Kode für LocalStorage
} else {
   // kein LocalStorage verfügbar
}
```

Einschränkungen des Local Storage

- Local Storage ist einfach gebrauchbar und damit auch einfach missbrauchbar. → Seien Sie vorsichtig!
- gemeinsam genutzt von allen Seiten im Browser
- Keine sensiblen Nutzerdaten im Local Storage speichern!
  (Verschlüsselung ist keine Lösung, da notwendiger Crypto-Kode mit Seiten ausgeliefert wird und zum Entschlüsseln verwendet werden kann)
- kein Ersatz für serverseitige Datenbank
- i.d.R. auf 5MB beschränkt
- Local Storage erlaubt nur synchrone Zugriffe

Local Storage: Beispiele

```
/* einfaches (K,V)-basiertes Speichern */
window.localStorage.setItem('name', 'Tenshi Hara');

/* Kompositstrukturen speichern mittels JSON */
const person = {
   name: "Tenshi Hara",
   funktion: "Dozent",
}
window.localStorage.setItem('user', JSON.stringify(person));

/* Lesen mittels Schlüssel
 * liefert hier {"name":"Tenshi Hara","funktion":"Dozent"} */
window.localStorage.getItem('user');
var nutzer = JSON.parse(window.localStorage.getItem('user'));

/* Löschen des o.g. (K,V)-Beispiels mittels Schlüssel */
window.localStorage.removeItem('name');

/* alles löschen */
window.localStorage.clear();
```

Session Storage

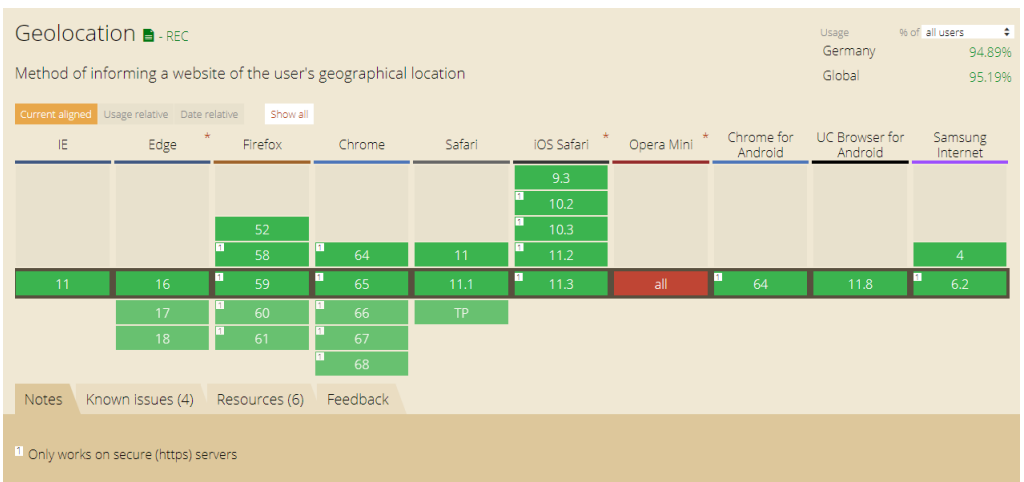
- Session Storage ist spezieller Local Storage
- hält Daten nur innerhalb einer Session
  (in den gängigen Browsern bedeutet dies: innerhalb **eines** Tabs (`_self`))
- bei Verlassen oder Duplizieren der Session gehen Daten verloren
  (insbesondere beim Folgen von Links außerhalb von `_self`)
- Handhabung äquivalent zum Local Storage, nur Bezeichnung ist anders:

```
window.sessionStorage
```

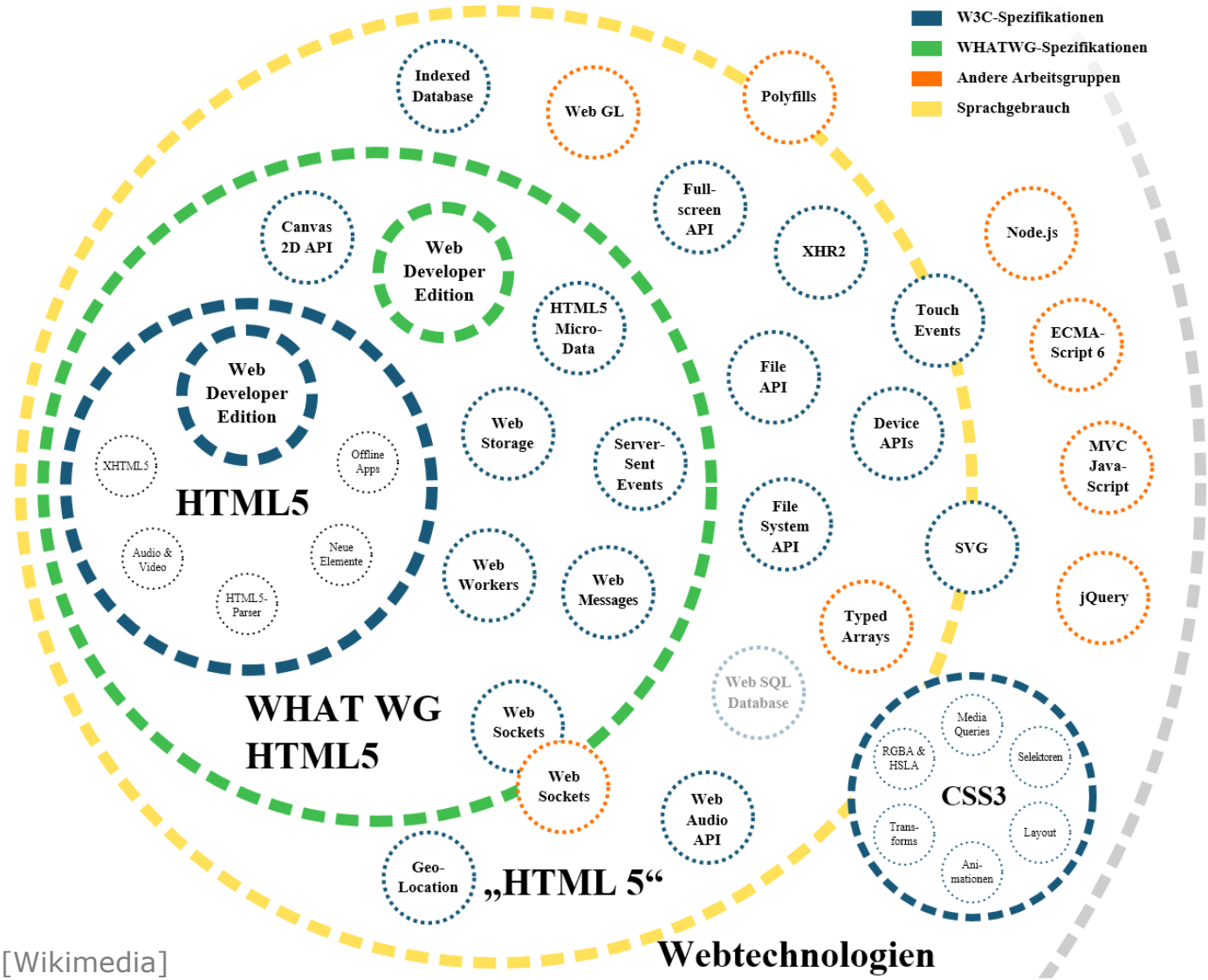
Zusammenfassung

HTML 5

- 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



HTML 5 im Sprachgebrauch



[Wikimedia]