Browser APIs

https://iuk.one/1066-1913

Clemens H. **Cap**
ORCID: 0000-0003-3958-6136

Department of Computer Science
University of **Rostock**
Rostock,Germany
clemens.cap@uni-rostock.de

Version 1

Zielstellung

Aktueller Stand (Februar 2023):

- 120 APIs mit separatem Namen und Standardisierungs-Komitee

Verbreitung neuer APIs:

- Vorne dabei: Chrome. Gelegentlich Firefox. Nachzügler: Safari

Bewertung:

- Manche APIs sehr sinnvoll.
- Manche APIs vermutlich Feature-war der Browser-Hersteller.
- Durchaus kritisch zu betrachtende Entwicklung
  Bloating, Browser Stabilität, Attack Surface, Fragmentierung

Ziel:

- Grober Überblick und Klassifikation
- Einmal davon gehört haben
- Anregung zur Nutzung

Processes and Tasks (1)

Background Tasks API:
Web Periodic Background Synchronization API:
Prioritized Task Scheduling API:

- Delegation spezifischer oder periodischer Aufgaben an einen Service Worker im Hintergrund.
- Abhängig von der jeweiligen Netz-Konnektivität.
- Priorisierung der Tasks möglich.
- Bsp: Synchronisation von offline lokal durchgeführten Edit Tasks (spezifisch)
- Bsp: Regelmäßiger download von News (periodisch)

Idle Detection API:

- Ermittlung von Zustände von Screen und User
- Screen-State: Locked oder unlocked
- User-State: Active oder idle
- Erlaubt Ereignis-basierte Programmierung statt Polling.

Processes and Tasks (2)

Screen Wake Lock API:

- Verhindert automatisches Screen Locking und Dimming
- Bsp: Navigation, eBook, Präsentationen

Network Information API:

- Ermittlung der Art der Netzwerk-Verbindung (Bsp: cellular, ethernet, bluetooth, wifi)
- Ermittlung der Qualität der Netzwerk-Verbindung (Bsp: Datenrate, Latenzzeit)
- Läuft System unter Nutzer-Anforderung einer verminderten Datenrate
- Ereignis-basierte Reaktion auf veränderte Netz-Anbindung

Worklets:

- Low-level Einstiegsmöglichkeiten in Rendering Pipeline
- In den Bereichen Audio, Animation und Layout vorgehalten
- Können in Javascript oder WebAssembly geschrieben sein.

Processes and Tasks (3)

Web Worker API:
Service Worker API:

- Erlaubt Hintergrund-Prozesse zur Bearbeitung von Tasks.
- Können unabhängig von Seite sein und Load Vorgang interceptieren.

Web Locks API: Scripts setzen Locks auf Ressourcen.

Web Assembly:

- Low level Assembler-artige Sprache für Browser.
- Native Performanz in spezieller Sandbox.

User Interface (1)

Keyboard API:

- Keyboard Mapping: Anpassung des Keyboard Layouts
- Keyboard Locking: Macht üblicherweise OS-relevante Tasten für web-Anwendungen verfügbar

VirtualKeyboard API: Standardisiert virtuelle Keyboards.

Gamepad API:

- Erlaubt Integration von Gamepad Controllern

Push API:
Notifications API:

- Ermöglicht Versenden von Server-Nachrichten an Background Seiten
- Ermöglicht Browser, OS-spezifische Benachrichtigungen zu erzeugen.
- Dient einer möglichst nahtlosen Integration von PWAs in das OS.
- Manche schöne Details (badges) von Apple nicht unterstützt.

User Interface (2)

Drag and Drop API:

- Erlaubt drag-and-drop Gesten innerhalb des Browsers
- Ermöglicht teilweise Anbindung an Bereiche außerhalb des Browsers
- Definiert etliche Events: `drag`, `dragend`, `dragenter`, `dragleave`, `dragover`, `dragstart`, `drop`
- Definiert visuelle Unterstützung und Daten-Transfer

Clipboard API:

- Erlaubt Browser Reaktion auf Clipboard Befehle (cut, copy, paste).
- Erlaubt Browser Zugriff auf System-Clipboard.
- Sicherheitstechnisch spannend (Bsp: Auslesen PW-Manager gepasteter Passwörter)

Share API: Zugriff auf OS spezifische Share API.

User Interface (3)

EyeDropper API:
- Tool um einzelne Pixels vom Bildschirm zu analysieren.

Contact Picker API:
- Zugriff auf den Contact Manager eines mobilen Endgeräts.
- Erst für Android verfügbar.

Vibration API:
- Ausspielen von “hapticals”
- Einzelne Vibrationen, Muster, Dauer-Vibration

Barcode Detection API:
- Erkennt Barcodes in Bild-artigen Objekten.
- API erlaubt leider nur polling, nicht interrupts

User Interface (4)

History API:

- Unterstützt `forward/back` Navigation.
- Erlaubt Manipulation der History.

Navigation API:

- Sehr neuer Nachfolger der History API.
- Erlaubt das Interceptieren und Umbiegen von Navigations-Handlungen des Users.
- Wichtig für Single Page Applications.
- Bsp: Navigation zu neuer URL wird übersetzt in ein Nachladen von Inhalten von neuer URL und Darstellen auf derselben Seite.

Launch Handler API:

- Welches Fenster wird bei einer Neuöffnung benutzt?
- Insbesondere: Browser versus lokal installierte PWA.
- Von Apple derzeit aus bekannten Gründen nicht unterstützt.

User Interface (5)

Fullscreen API:

- Browser nutzt vollen Screen
- Sicherheitsproblem: Wie unterscheidet Nutzer Webseite von Browser-Elementen
- Beispiel: Web-Spoofing Attacke zeigt manipulierte Location-Zeile an.
  Nutzer ist auf anderer Seite, als er glaubt.
- Darf daher nur durch Benutzer-Geste aktiviert werden.
- Fullscreen Status muß leicht ersichtlich und einfach zu verlassen sein.

Window Controls Overlay API: Zugriff auf OS-seitige Window Bedienelemente.

Device Orientation API:
Screen Orientation API:

- Zugriff auf Beschleunigungs-Sensor
- Ermittelt Beschleunigung (mit und ohne Gravitation)
- Ermittelt Rotations-Raten um 3 Raumachsen.
- Unterscheidet Landscape und Portrait Orientierung.
- Detektiert Gesten (Bsp: Schütteln, Kippen, Balancieren)

User Interface (6)

Content Index API: Überblick über offline-verfügbare Inhalte im ServiceWorker

Local Font Access API: Zugriff auf lokal installierte Fonts.

Beachte:

- Je mehr lokalen Status ein browser hat (local fonts, local media devices, local content)
- umso leichter fällt User tracking.
- Google Chrome unterstützt hohe Anzahl statushaltiger APIs: Warum?

User-Agent Client Hints API:

- User Agent ist etwa: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.107 Safari/537.36
- Übermittelt Browser Infos über Client Fähigkeiten
- Problem: Tracking
- Diese API erlaubt statt String spezifischere Abfragemöglichkeit
- Gestattet auch Abgrenzung nach übertragener Tracking Entropie
- Ist damit Privatheits-freundlicher.

User Interface (7)

Ink API:

- Anwendung: Zeichnen und schreiben via Stift am Browser
- Problem: Latenz zu hoch, Event Frequenz zu gering da Touch Events durch Event Loop gehen
- Lösung 1: Libraries, die Punkte glatt verbinden.
  Bsp: https://github.com/steveruizok/perfect-freehand
  Demo: https://perfect-freehand-example.vercel.app/
- Restproblem: Latenz
- Lösung 2: Ink API liefert shortcut durch die Event Loop
- Demo: https://mabian-ms.github.io/delegated-ink-trail.html

User Interface (8)

Touch Events API:
Pointer Events API:
UI Events API:

- Finger, Trackpad, SmartPen usw. sind komplexer als die Maus.
- Emulation ist nicht immer möglich oder im OS schlecht realisiert.
- Anpassung an dieser Geräte, abstraktere und zusammengesetzte Events.
- Ziel: Unified Interface.

Pointer Lock API:

- Bsp: Will Objekt mit Pointer verschieben
  User bewegt Pointer so schnell, daß Verbindung zum Objekt verloren geht
- Pointer Lock API erlaubt dauerhafte Verbindung zum Event Target
- Wenn schlampig programmiert kann User Pointer "verlieren".

User Interface (9)

Popover API:

- Popovers bisher nur als rudimentäre Modals (`alert`), mit HTML Mitteln oder Libraries (bbotstrap).
- Sicherheitstechnisch heikel (Bsp: Verdecken von Warnungen)
- Hier jetzt Standardisierung der Popovers.
- Sehr neue API, auch von apple unterstützt.
- Entlastung des Web von der Popover Seuche.
- Mutmaßlich: Neue Werbeformate.

Selection API: Auswahl von Dokumententeilen durch User.

Visual Viewport API: Dynamische Anpassung des Viewport.

User Interface (10)

Presentation API:

- Möglichkeit, Präsentationen auf externem Bildschirm zu zeigen.
- Leider stark eingeschränkt
- Target schein Google Chromecast zu sein.

Window Management API:

- Erlaubt Positionierung von Fenstern auf extern angeschlossene Monitore.
- Dort dann kaum mehr Restriktionen.
- Besser für Präsentationen geeignet als Presentation API!

Web Speech API:

- Text-to-Speech und Speech-to-Text im Browser.
- Demo: https://mdn.github.io/dom-examples/web-speech-api/speech-color-changer/

Developer APIs (2)

Console API:

- Zugriff auf die logging Console.
- Im Laufe der Zeit immer wieder ergänzt.
- assert, clear
- Zähler von ereignissen
- Nachrichten unterschiedlicher Dringlichkeit
- Ausgabe linearisierter Objekte
- Ausgabe von Objekten als aufklappbare Struktur
- Ausgabe von Zeitstempeln
- Ausgabe von Nachrichten mit zusätzlichem Zeitstempel
- Gruppen und kollabierte / kollabierbare Gruppen
- Tabellen-Formatierung der Ausgabe
- Ausgabe eines Stack Trace.
- Erzeugen eines Performance Profiles.

Developer APIs (2)

Performance API:

- Besteht aus einer Vielzahl von Sub APIs.
- Erlaubt das Vermessen und Erkennen der Browser Performance.
- Viele Bereiche: Rendering, Server, User, Kommunikation, Layout Loops, etc.
- Heute nur mehr wichtig für high performance Seiten
  Bsp: Flugsimulator im Browser

URL API:

URL Pattern API:

- Unterstützt Analyse der URL Strukturen
- Unterstützt clientseitiges Application-Routing durch URL Patterns

Device APIs (1)

Battery Status API

- Ermittelt Zustand der Stromversorgung
- Ereignisse bei Änderungen der Stromversorgung
- Erlaubt Reaktion der Web Anwendung auf Stromversorgungslage

Bluetooth API:

- Verbindung mit einem Bluetooth Low Energy (BLE) Gerät
- Einige (wenige) Interaktionsmöglichkeiten mit dem BLE Gerät

Geolocation API

- Anbindung an GPS und Wifi SSID Location Dienste
- Wichtige Sicherheits-Implikationen

Device APIs (2)

Web NFC API:

- Wenig verbreitet und erst extrem kurz verfügbar
- Zugriff auf NFC tags

Sensor API:

- Anbindug generischer Sensorik
- Bsp: Gyroskop, Proximity, Umgebungslicht, Magnetometer, und mehr.
- Nicht von Apple unterstützt.

Web HID API:
WebUSB API:

- Einbinden weiterer Peripherie-Geräte in die Web-Nutzung

Media APIs (1)

MediaStream Image Capture API:

- Erfaßt Bilder und Videos von cameras
- Unterstützt treiberabhängig wenige native Eigenschaften der Engeräte
  Bsp: Blitz, Red-Eye reduction

Audio Output Devices API:

- Erlaubt Webpage die Beeinflussung der Auswahl der genutzten Audio Ausgabe Geräte.
- Beispiel: Audio Routing an Headset oder Lautsprecher.

Media Capture and Streams API:

- Allgemeine API für streaming Audio und Video.

MediaStream Recording API:

- erweitert Media Capture and Streams API um client-seitiges Recording.

Media APIs (2)

Media Source API:
Media Source Extensions API:

- Erlaubt das Fortsetzen von Media-Strömen aus anderen, wechselnden Quellen.
- Voraussetzung für DASH (dynamic adaptive streaming over http).
- Bsp: Wechsel in Datenrate erlaubt nahtlosen Wechsel auf andere Video-Codierung
- Bsp: Dynamisches Einschneiden von Werbung in YouTube Videos.

Encrypted Media Extensions API:

- Erlaubt das kopierverhindernde (DRM) Abspielen verschlüsselter Medien.
- Problem: Erfordert closed source Plug-Ins in open source Browsern.
  Führt auf grundsätzliche Widersprüchlichkeiten.

Media APIs (3)

WebCodecs API:

- Zugang zu den Internas der Video und Audio Codecs.
- Beispiel: Einzelne Frames im Video
- Für Web-basierte Video und Audio Editoren

WebVTT: Timed Text im Video Player.

Media Capabilities API:

- Stellt Informationen bereit über vorhandene Media-Unterstützung und Codierungen

Picture in Picture API:

- Erlaubt Programmierung frei flottierender Video Fenster oberhalb der übrigen Web-Seite
- Automatisiert damit ein beliebtes (Mac)OS Feature.

Screen Capture API: Unterstützt (Teil-)Screenshots im Browser.

Web MIDI API: Zugang zu MIDI Datenströmen

Kommunikations API (1)

XMLHttpRequest API: Bekannt

Fetch API:

- Sehr reichhaltige API für http Anfragen
- Ersatz für XMLHttpRequest
- Durchgängig Promise-orientiert
- Zugriff auf Header
- Zugriff auf alle `request` und `response` Details
- Ausgeklügelte Parser für die Ergebnisse
- Erlaubt auch Behandlung von Streaming-Situationen

Streams API: Konzepte für Stream-Verarbeitung.

Kommunikations API (2)

```
async function getUsers() {
  let url = 'users.json';
  try {
    let res = await fetch(url);
    return    await res.json();
  } catch (error) {console.log(error);}
}
```

Src. 1: Beispiel eines JSON-Abrufs inklusive ergebnis-Behandlung mit Fetch API

Kommunikations API (3)

```
1  fetch("./tortoise.png")        // fetch image
2    .then((response) => {        // retrieve response
3      const reader = response.body.getReader(); // format body as reader
4      return new ReadableStream({
5        start(controller) {
6          return pump();
7          function pump() {
8            return reader.read().then(({ done, value }) => {
9              if (done) {controller.close(); return;}   // If done: close the stream
10             controller.enqueue(value); // Enqueue the next data chunk
11             return pump(); });  }  },  });  })
12   .then((stream) => new Response(stream))        // Create new response from stream
13   .then((response) => response.blob())           // Convert to blob
14   .then((blob) => URL.createObjectURL(blob))     // Convert to object url
15   .then((url) => console.log((image.src = url)))
16   .catch((err) => console.error(err));           // Error handler important
```

Src. 2:

Kommunikations API (4)

WebSocket API: Seite kann Socket Verbindung anbieten / öffnen zu Server.

WebTransport API:

- Erweiterung der WebSocket API.
- implementierung des HTTP/3 Protokolls.
- Erlaubt mehrere parallele Kommunikationskanäle.
- Erlaubt Unidirektionale Kommunikationskanäle.
- Erlaubt out-of-order Nachrichten.
- Erlaubt UDP-artige Kommunikation.

Kommunikations API (5)

ChannelMessaging API:

- Ermöglicht Kommunikation mit anderen Browser Kontexten.
- Konstruiert dazu Punkt-zu-Punkt Kanal-Konzepte.
- Geht nur minimal über `postMessage` hinaus.

Broadcast Channel API:

- Ermöglicht Kommunikation mit anderen Browser Kontexten.
- Benötigt keine Referenz auf den Partnern sondern geht mit Kanalkonzept
- Kanäle erlauben eine art publish-subscribe Konzept.

Beacon API:

- Sendet asynchrone, nicht-blockierende Requests an den Server.
- Keine Antwort vom Server erwartet.
- Kommt mit Garantie, daß Nachricht bei einem Seitenaufruf abgesendet wird
- Bsp: Zählen von Seiten-Aufrufen, Analytics Aufgaben
- Bsp: Server-seitigen Zustand einer Sitzung managen

Kommunikations API (6)

Compression Streams API:

- gzip Funktionalität als Teil einer API, nicht einer Library
- Ziemlich unaufgeregt.

WebRTC API:

- Sehr reichhaltige API mit vielen Erweiterungen
- Audio, Video, Data-Transport
- Erlaubt P2P Verbindungen, auch durch NAT Gateways.

Storage APIs (1)

File API:
File System Access API:
File and Directoriy Entries API:

- Gestattet den Zugriff auf das lokale Filesystem.
- Streng gebunden an Benutzer-Transaktionen.
- Bsp: Drag-and-drop, File-chooser, Clipboard.
- Enthält auf ein “Save” Interface.
- Bietet perspektivisch auch privates sandboxed Filesystem an.
- Hat diverse Einschränkungen
  Etwa: Zahl der Files die von Verzeichnissen zugänglich sind

Storage APIs (2)

IndexedDB API:

- Low-level API für key-value Datenbank
- Ermöglicht Transaktionen
- Unterstützt Cursor und Range Konzepte, Indizes und mehr
- Nicht einfach zu benutzen – Kapselungs-Bibliotheken sind sinnvoll
- Dexie.js: Einfacher, hilfreicher Wrapper
- JsStore.js: Bietet SQL-artige Schnittstelle

Server-seitige SQL Schnittstelle:

- knex.js
  Anbindung zu PostgreSQL, MSSQL, MySQL, SQLite3, Oracle und Redshift

WebSQL: Sqlite in the browser. Leider deprecated.

Storage APIs (3)

document.cookie

localStorage, sessionStorage

Quota API

Cookie Store API:

- Erlaubt Zugriff auf Cookies.
- Geht deutlich weiter als `document.cookie`
- Ermöglicht dem ServiceWorker Zugriff auf Cookies
- Gestattet asynchronen Zugriff auf Cookies außerhalb der Event Loop
- Erlaubt Events bei Änderung von Cookies.
- Noch in Entwicklungsphase.

DOM API

HTML DOM API:

- Sehr große API.
- Realisiert die W3C DOM (document object model) Strukture.
- Umfaßt auch eine sehr reichhaltige SVG API für Vektorgrafik.

HTML Sanitizer API:

- Sanitizer verarbeiten im Browser syntaktisch inkorrektes HTML.
- Bisher nur Browser-intern verfügbar.
- API macht das nun dem JS Programmierer zugänglich.

CSS

Es bestehen eine ganze Reihe von Schnittstellen auf CSS.

- CSSOM – analog dem DOM.
- Zugriff auf CSS, Style-Vorschreibungen und genutztem Style.
- Dynamisches Laden von Fonts.
- Highlighten beliebiger Textabschnitte.
- CSS-Parser (in Entwicklung)
- Font-Metrik (in Entwicklung)
- Geometrische Objekte und Transformationen

CSS/DOM–nahe APIs

Intersection Observer API:

- Überlappung von Elementen lief bisher in Javascript auf dem Main Thread
- Blockiert damit Main Thread
- Auslagerung in Worker geht aber nicht (document Zugriff)
- Lösung durch Intersection Observer API.

Resize Observer API: Ganz ähnlich.

Page Visibility API: Unterstützt Sichtbarkeitsanalyse ganzer Seiten.

Web Animations API:

View Transition API:

- Animationen und Übergänge zwischen Sichten.

Web Components API:

- Customs: Eigene HTML Tags mit CSS und JS Verhalten
- Shadows: Kapseln von Elementen nach außen zum Schutz des Verhaltens
- Templates: Keine Template Engine in HTML

User Identification (1)

Credential Management API:
Federated Credential Management API:
Web Authentication API:

- Verwaltet Passwörter, föderierte Credentials (Bsp: OpenId) und lokale Credentials (Fingerprint)

Web Crypto API:

- Stellt Crypto Aufrufe zur Verfügung.
- Achtung: Nur durch Spezialisten sinnvoll verwendbar.
- Achtung: Ohne Audit weiß man nicht, ob die Webseite das auch richtig macht.

Web OTP API:

- Stellt für Server sicher, daß Benutzer eine bestimmte Telephonnummer hat
- Reagiert auf eine spezifisch formatierte SMS mit Erzeugung eines One Time Password

User Identification (2)

Payment Request API:
Payment Handler API:

- Unterstützt Zahlungsvorgänge im Browser
- Speichert vorhandene / autorisierte Zahlungsinstrumente
- Unterstützt redirect auf sichere Payment Gateways
- Versucht, die sicherheitstechnisch heikle Anbindung an Zahlvorgänge für den Entwickler zu vereinfachen und zu standardisieren

Permissions API:

- Versuch, diverse Mechanismen der User-Berechtigung zu standardisieren
- Bsp: Clipboard, Notifications, Push, MIDI
- Zunehmend sollen mehr APIS eingebunden werden.

Graphik

Canvas API:

- Pixelgrafik (im Gegensatz zu Vektorgrafik)
- Erlaubt 2D Zeichen-Operationen
- Kein Szene-Graph

WebGL API: Zugang zu OpenGL ES 2.0 und 3.0

WebGPU API:

- Sehr sehr neu.
- Zugang zur GPU für rechenintensive Aufgaben.

WebVR API: und **WebXR Device API:** Virtual Reality Devices

Sonstige

Encoding API:

- Für legacy und non-UTF-8 Zeichencodierungen.

Anhang

Übersicht

Programmquellenverzeichnis

Rechtliche Hinweise (1)

Die hier angebotenen Inhalte unterliegen deutschem Urheberrecht. Inhalte Dritter werden unter Nennung der Rechtsgrundlage ihrer Nutzung und der geltenden Lizenzbestimmungen hier angeführt. Auf das Literaturverzeichnis wird verwiesen. Das Zitatrecht in dem für wissenschaftliche Werke üblichen Ausmaß wird beansprucht. Wenn Sie eine Urheberrechtsverletzung erkennen, so bitten wir um Hinweis an den auf der Titelseite genannten Autor und werden entsprechende Inhalte sofort entfernen oder fehlende Rechtsnennungen nachholen. Bei Produkt- und Firmennamen können Markenrechte Dritter bestehen. Verweise und Verlinkungen wurden zum Zeitpunkt des Setzens der Verweise überprüft; sie dienen der Information des Lesers. Der Autor macht sich die Inhalte, auch in der Form, wie sie zum Zeitpunkt des Setzens des Verweises vorlagen, nicht zu eigen und kann diese nicht laufend auf Veränderungen überprüfen.

Alle sonstigen, hier nicht angeführten Inhalte unterliegen dem Copyright des Autors, Prof. Dr. Clemens Cap, ©2020. Wenn Sie diese Inhalte nützlich finden, können Sie darauf verlinken oder sie zitieren. Jede weitere Verbreitung, Speicherung, Vervielfältigung oder sonstige Verwertung außerhalb der Grenzen des Urheberrechts bedarf der schriftlichen Zustimmung des Rechteinhabers. Dieses dient der Sicherung der Aktualität der Inhalte und soll dem Autor auch die Einhaltung urheberrechtlicher Einschränkungen wie beispielsweise Par 60a UrhG ermöglichen.

Die Bereitstellung der Inhalte erfolgt hier zur persönlichen Information des Lesers. Eine Haftung für mittelbare oder unmittelbare Schäden wird im maximal rechtlich zulässigen Ausmaß ausgeschlossen, mit Ausnahme von Vorsatz und grober Fahrlässigkeit. Eine Garantie für den Fortbestand dieses Informationsangebots wird nicht gegeben.

Die Anfertigung einer persönlichen Sicherungskopie für die private, nicht gewerbliche und nicht öffentliche Nutzung ist zulässig, sofern sie nicht von einer offensichtlich rechtswidrig hergestellten oder zugänglich gemachten Vorlage stammt.

Use of Logos and Trademark Symbols: The logos and trademark symbols used here are the property of their respective owners. The YouTube logo is used according to brand request 2-9753000030769 granted on November 30, 2020. The GitHub logo is property of GitHub Inc. and is used in accordance to the GitHub logo usage conditions https://github.com/logos to link to a GitHub account. The Tweedback logo is property of Tweedback GmbH and here is used in accordance to a cooperation contract.

Rechtliche Hinweise (2)

Disclaimer: Die sich immer wieder ändernde Rechtslage für digitale Urheberrechte erzeugt ein nicht unerhebliches Risiko bei der Einbindung von Materialien, deren Status nicht oder nur mit unverhältnismäßig hohem Aufwand abzuklären ist. Ebenso kann den Rechteinhabern nicht auf sinnvolle oder einfache Weise ein Honorar zukommen, obwohl deren Leistungen genutzt werden.

Daher binde ich gelegentlich Inhalte nur als Link und nicht durch Framing ein. Lt EuGH Urteil 13.02.2014, C-466/12 (Pressemitteilung, Blog-Beitrag, Urteilstext). ist das unbedenklich, da die benutzten Links ohne Umgehung technischer Sperren auf im Internet frei verfügbare Inhalte verweisen.

Wenn Sie diese Rechtslage stört, dann setzen Sie sich für eine Modernisierung des völlig veralteten Vergütungs- und Anreizsystems für urheberrechtliche Leistungen ein. Bis dahin klicken Sie bitte auf die angegebenen Links und denken Sie darüber nach, warum wir keine für das digitale Zeitalter sinnvoll angepaßte Vergütungs- und Anreizsysteme digital erbrachter Leistungen haben.

Zu Risiken und Nebenwirkungen fragen Sie Ihren Rechtsanwalt oder Gesetzgeber.

Weitere Hinweise finden Sie im Netz hier und hier oder hier.

Zitierweise dieses Dokuments

Wenn Sie Inhalte aus diesem Werk nutzen oder darauf verweisen wollen,
zitieren Sie es bitte wie folgt:

Clemens H. Cap: Browser APIs. Electronic document. https://iuk.one/1066-1913 5. 7. 2023.

Bibtex Information: https://iuk.one/1066-1913.bib

```
@misc{doc:1066-1913,
   author       = {Clemens H. Cap},
   title        = {Browser APIs},
   year         = {2023},
   month        = {7},
   howpublished = {Electronic document},
   url          = {https://iuk.one/1066-1913}
}
```

Typographic Information:
Typeset on ?today?
This is pdfTeX, Version 3.14159265-2.6-1.40.21 (TeX Live 2020) kpathsea version 6.3.2
This is pgf in version 3.1.5b
This is preamble-slides.tex myFormat©C.H.Cap

Verzeichnis aller Folien

Legende:
- Fortsetzungsseite
- Seite ohne Überschrift
- Bildseite