openHPI: **Web-Technologien**

Willkommen zu Kurswoche 3

Prof. Dr. Christoph Meinel

Hasso-Plattner-Institut

Universität Potsdam

Einführung

Begriff der **Web-Programmierung** ist sehr weit gefasst.
Zur Web-Programmierung gehören

- Client- und Serverseitige Programmierung
- Middleware-Architekturen
- Web-Services

Behandeln **diese Woche** folgende **Konzepte der Web-Programmierung**:

- DOM – Document Object Model
- HTTP-/WWW-/Anwendungs-Server
- Sessions
- Informations-/Zeichenkodierung
- Grafik-/Audio-/Video-Kodierung

Behandeln in **Woche 4**:

Clientseitige Web-Programmierung in der Praxis

- JavaScript
- Wichtige Frameworks und Bibliotheken
- AJAX: Asynchrones JavaScript und XML
- Programmiergrundlagen

Behandeln in **Woche 5**:

Serverseitige Web-Programmierung in der Praxis

- CGI-Interface
- Technologien und Programmiersprachen
- Web-Frameworks
- Web-Services
- Programmieren mit Ruby / Ruby on Rails

openHPI: **Web-Technologien**

Web-Programmierung

Prof. Dr. Christoph Meinel

Hasso-Plattner-Institut

Universität Potsdam

Begriff der **Web-Programmierung** ist sehr weit gefasst.

Zur Web-Programmierung gehören

- Client- und Serverseitige Programmierung
- Middleware-Architekturen
- Web-Services

Motivation

- Ursprünglich statische Webseiten werden immer lebendiger und inhaltlich komplexer:
  - Mash-Ups / Widgets
    - Einbindung, Vernetzung von Inhalten anderer Webseiten
  - Interaktive Webseiten
    - Nutzerinhalte, Browser als Betriebssystem
- **Beispiele**:
  - Online-Karten
  - Office-Anwendungen
  - Grafik-intensive Anwendungen, z.B. Spiele im Browser

➔ **Je interaktiver, umso mehr Web-Programmierung**

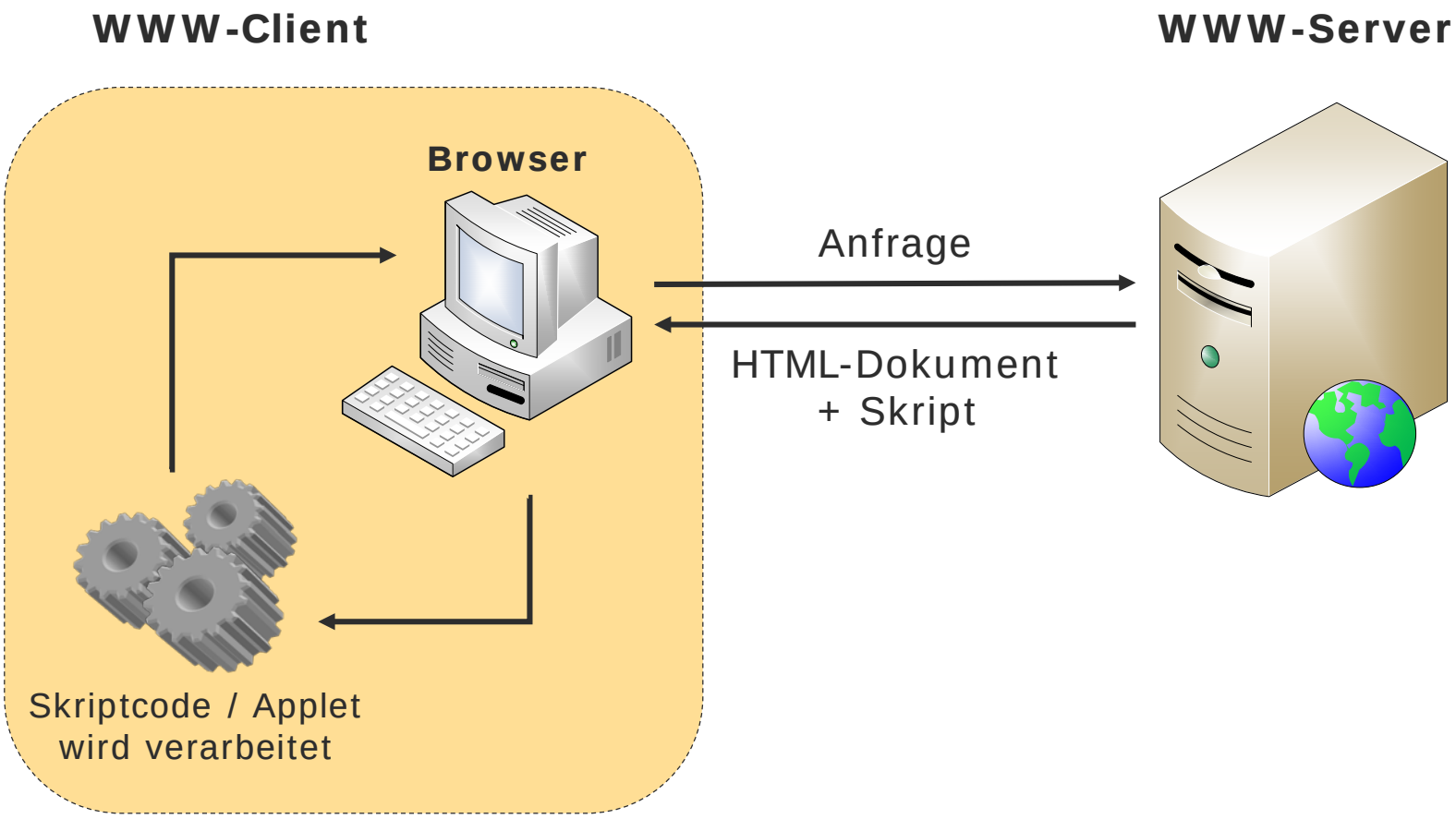
Clientseitige Web-Programmierung (1/2)

Hier erfolgt interaktive Ausgestaltung der Website auf Client-Seite

Ausführungsmodell

- Ausführbarer/interpretierbarer Programmcode wird vom WWW-Server zusammen mit angefordertem HTML-Dokument an WWW-Browser (Client) übergeben und auf Client-Rechner ausgeführt
- WWW-Client benötigt dazu Interpreter (ggfs. als Plugin), der übergebenen Programmcode interpretieren und zusammen mit dem WWW-Client ausführen kann
  - JavaScript-Programme (bzw. -Skripte)
  - Java-Applets
  - Microsoft Silverlight und Adobe Flash

Ausführung im Browser



Serverseitige Web-Programmierung (1/4)

Hier wird Website interaktiv und dynamisch auf Server-Seite erstellt

Motivation

- Server-Seite übernimmt:
  - Speichern und Abruf von Daten aus Datenbanken
  - Kommunikation mit anderen Web-Services
  - Speichern von nutzergenerierten Inhalten
  - Versenden von Emails und Benachrichtigungen
  - Validierung von Daten
  - Ausliefern von HTML

➔ **Server speichert und transformiert den Zustand der gesamten Datenbasis**

Serverseitige Web-Programmierung (2/4)

Bei Browserabruf können HTML-Dokumente:

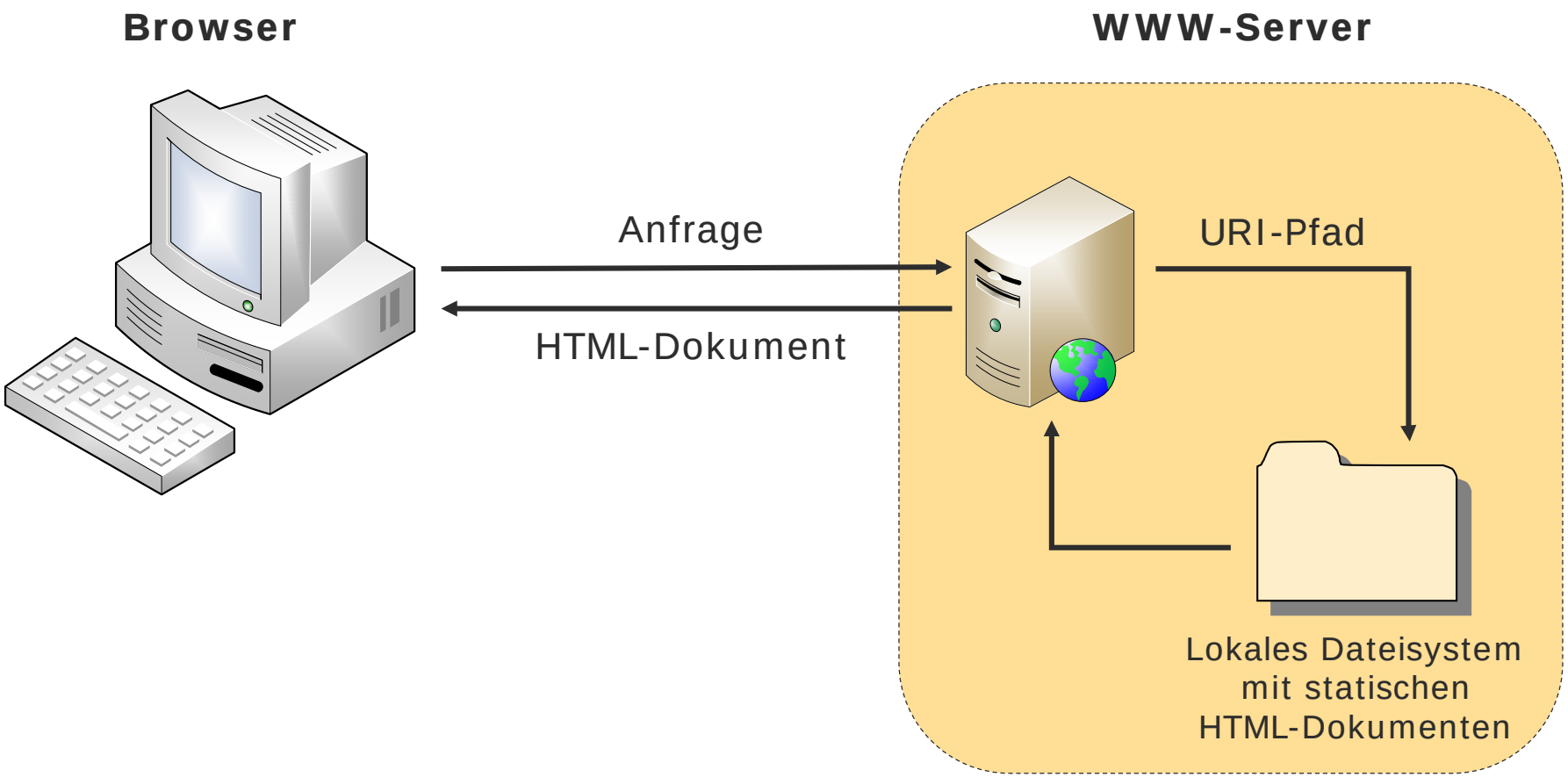
- **statisch** vorliegen oder
- **dynamisch** – erst auf die Browser-Abfrage hin – erzeugt/generiert werden

bevor sie durch WWW-Server ausgeliefert werden

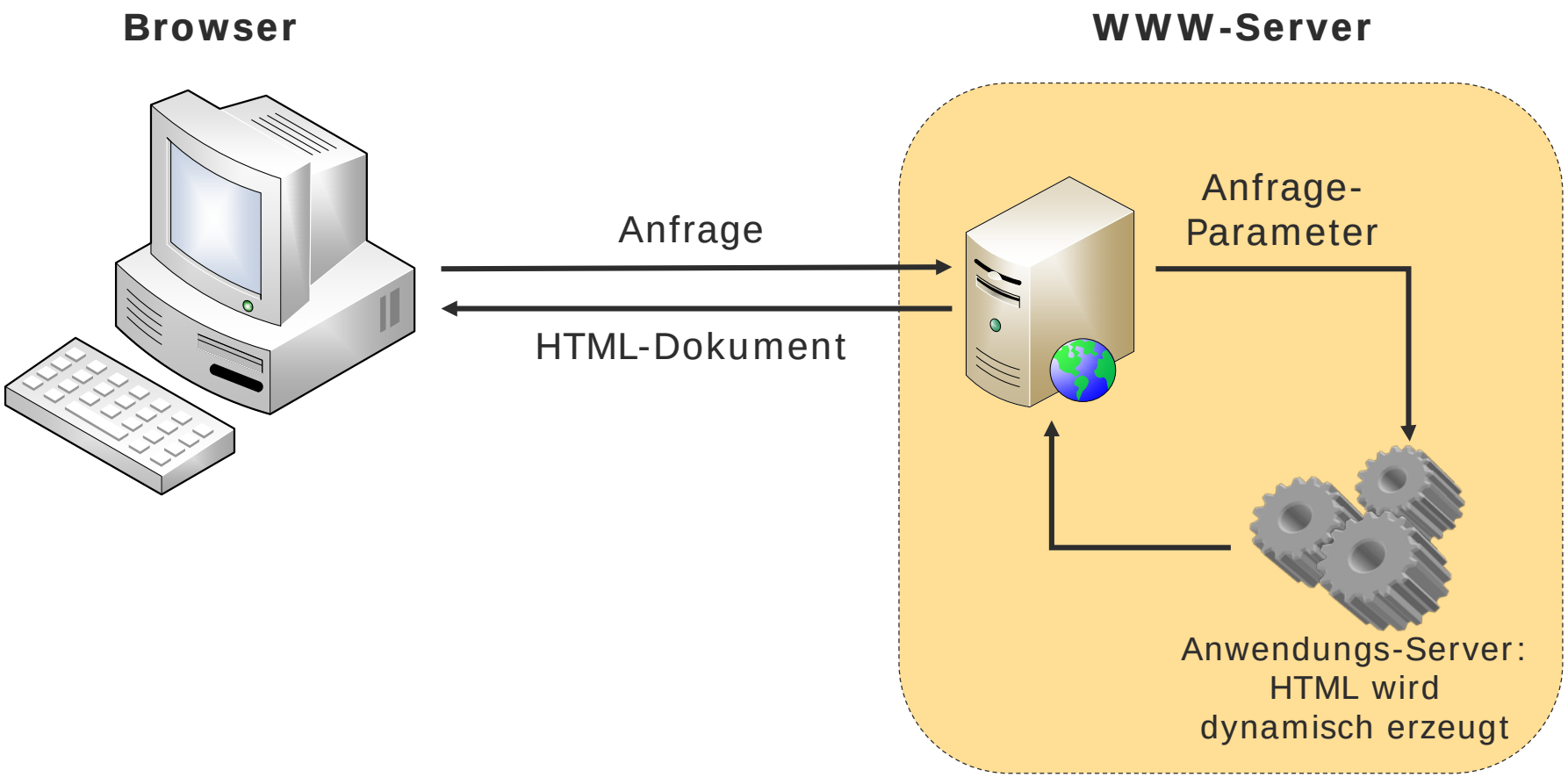
Dynamisch Erzeugung sinnvoll, z.B. bei

- nutzergenerierten Inhalten oder
- wenn Anfragen aus Datenbank beantwortet werden müssen, z.B.:
  - Warenkatalogen, z.B. amazon.de
  - Suchmaschinen, z.B. google.de
  - Sozialen Netzwerken, z.B. facebook.com
  - Zeitungen, z.B. heise.de

Statische Webseiten



Dynamische Webseiten



openHPI: **Web-Technologien**

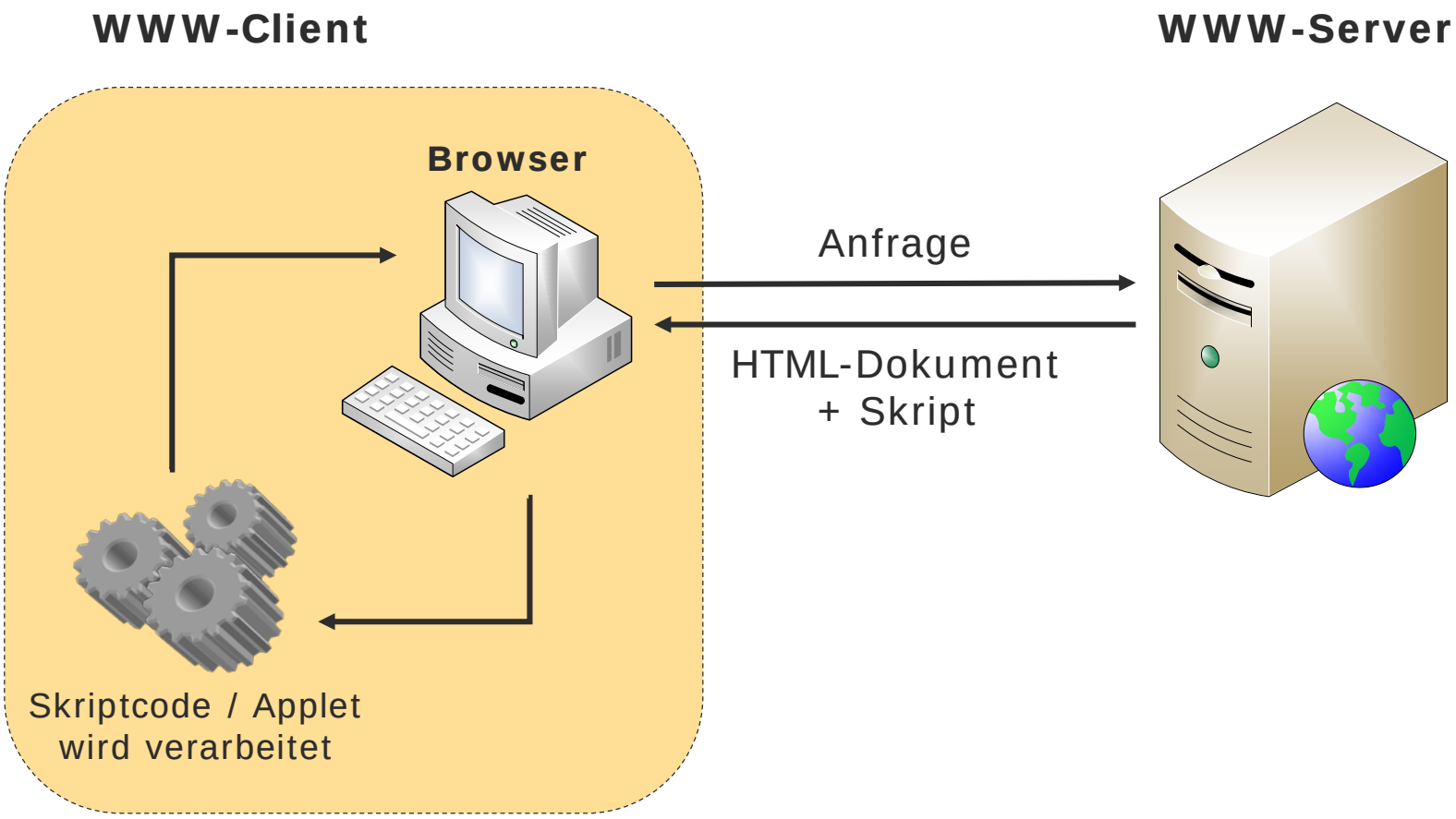
Clientseitige Web-Programmierung

Prof. Dr. Christoph Meinel

Hasso-Plattner-Institut

Universität Potsdam

Ausführung im Browser



Laden von Skripten

- Skript-Dateien werden in HTML-Beschreibung der Website referenziert
- Lösen zusätzlichen HTTP-Request zum Server aus, um das Skript zu laden
- Skripte: erst parsen, dann ausführen

- Laden und Ausführen von Skripten blockiert Anzeige von restlichem HTML

  ➔ Skripte am Ende einbinden

- Anzahl und Größe der Skripte haben Auswirkung auf die Geschwindigkeit der Webseite

```
<html>
  …

  <body>
    …

    <script
      type="text/javascript"
      src="script.js"
    >
    </script>
  </body>
</html>
```

JavaScript

JavaScript: Einzige weit verbreitete und **standardisierte** Technologie für clientseitige Skripte in Webseiten

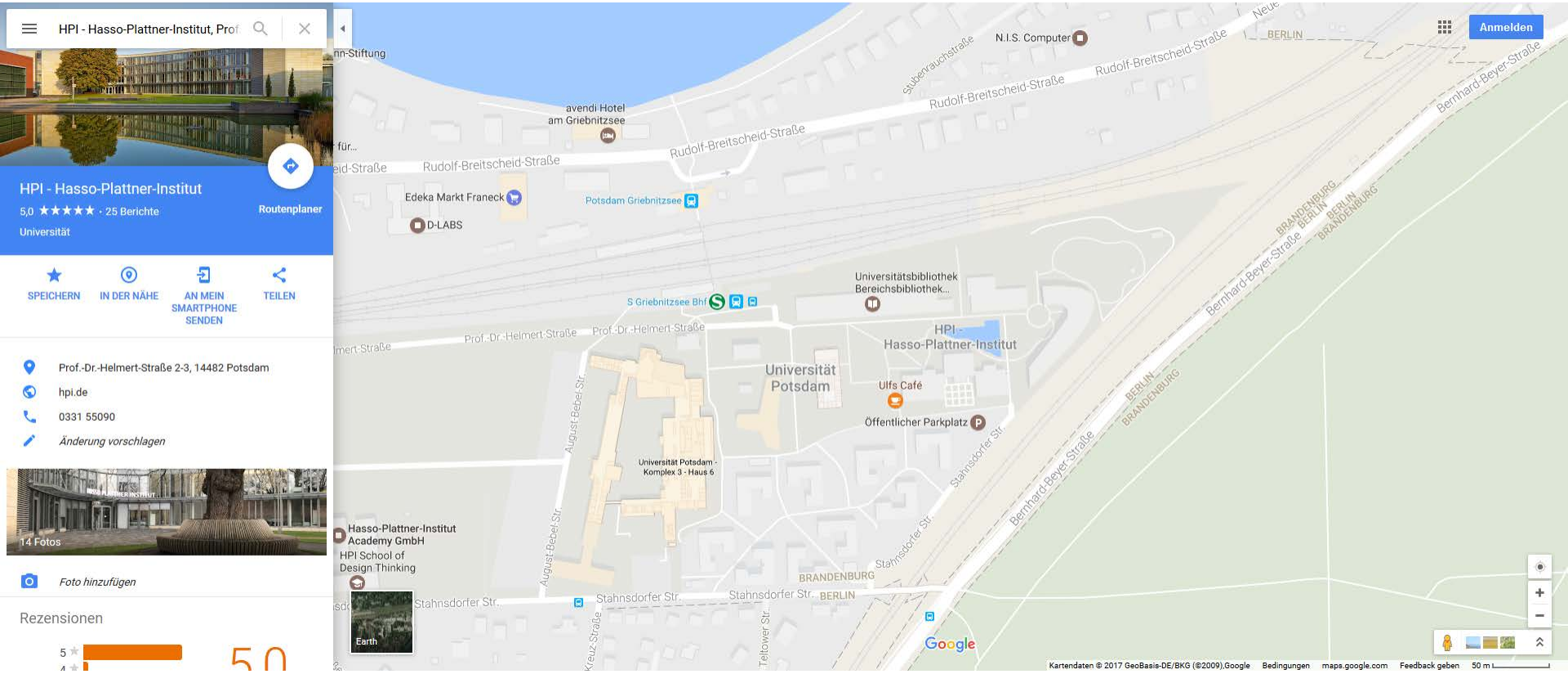
Features:

- Zugriff auf & Manipulation der HTML-Struktur möglich: **Document Object Model (DOM)**
- Reaktion auf vom Nutzer ausgelöste **Events**
- Asynchrone Server-Kommunikation im Hintergrund: **AJAX**, **Websockets**
- Animationen, 3D-Grafik
- Zugriff auf Ort, Batteriestatus, Webcam, Mikrofon, Bewegungssensoren, …
  - Inspiriert vom Funktionsumfang mobiler Apps
- **Service Worker**: ermöglicht Offline-Zugriff auf Webseiten
- u.v.m.

- Mehr dazu in **Woche 4** (Funktionsumfang, Programmier-Tutorial)

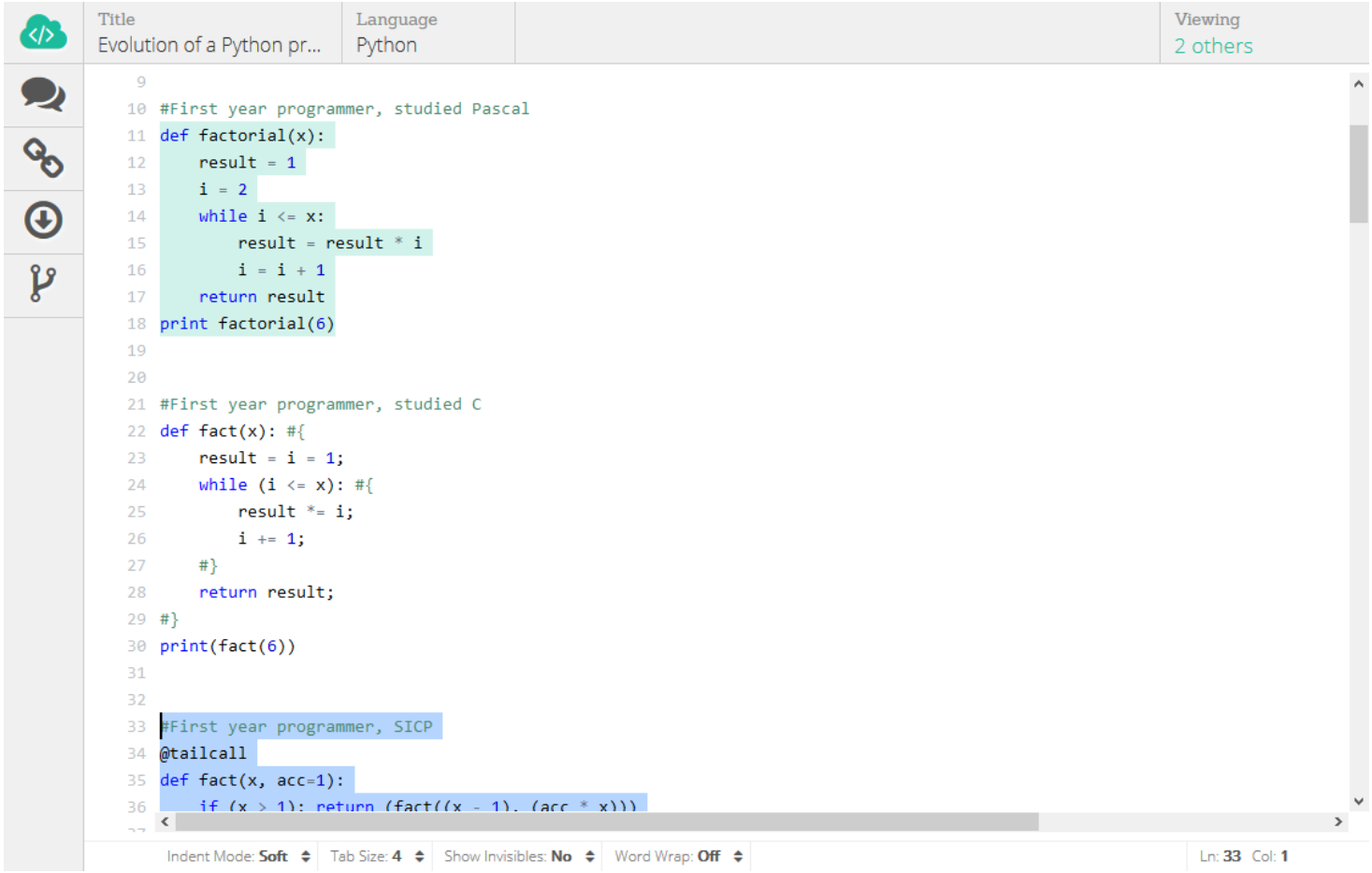
JavaScript: Anwendungsbeispiele (1/3)

Online-Karten



JavaScript: Anwendungsbeispiele (2/3)

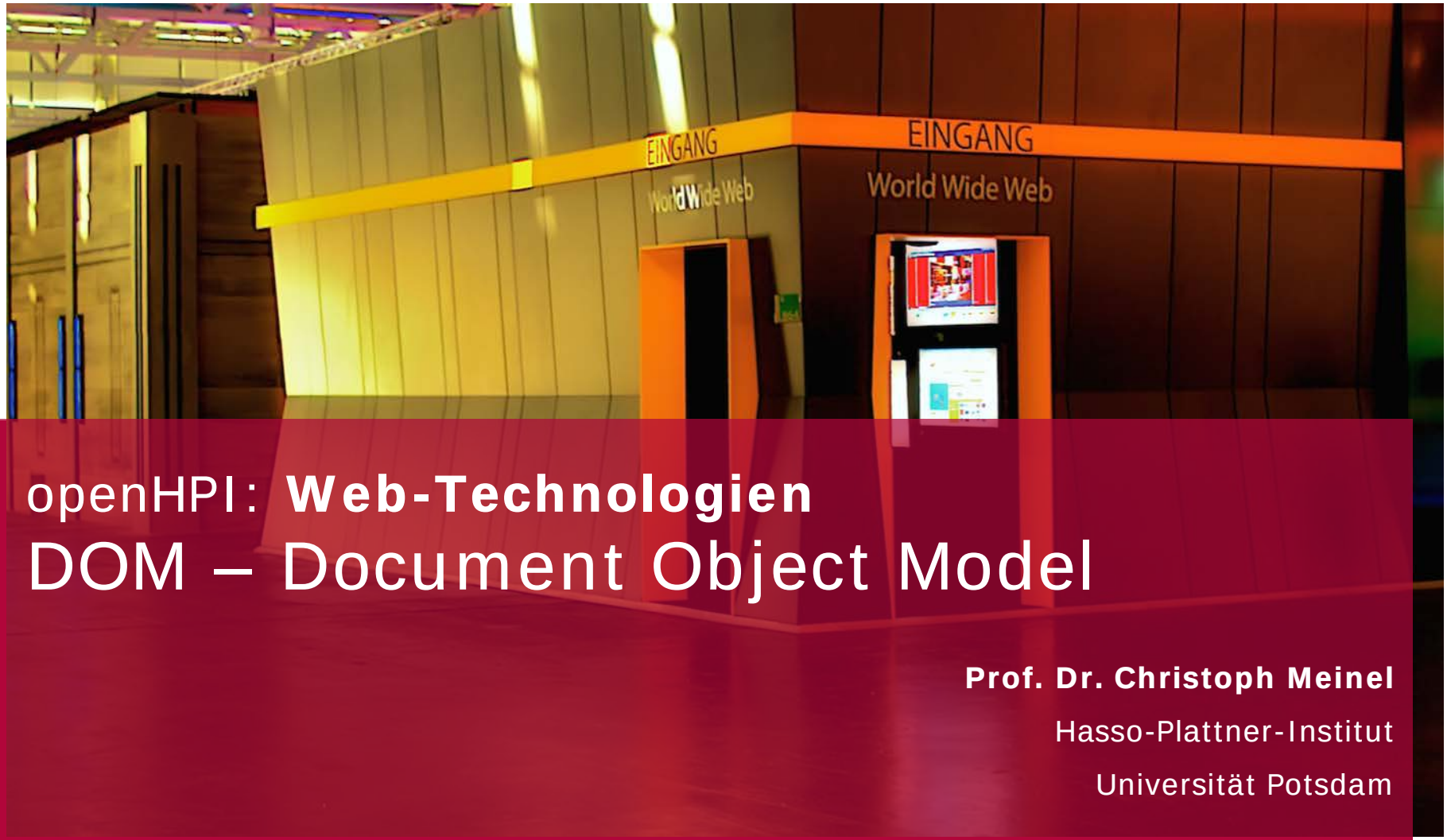
Kollaborations-Tools



Browser-Spiele

Flash, Silverlight, Java-Applets

- Haupttreiber für die **ursprüngliche** Verbreitung dieser Technologien:
  - Fehlende Unterstützung von Audio und Video im Browser
  - Schlechte Performance von JavaScript
  - Fehlende Features in den Standard-Technologien des Webs
- Und nun?
  - Audio und Video: seit HTML5 nativ unterstützt
  - Verbesserte Geschwindigkeit und Funktionsumfang von JavaScript
  - Animationen, interaktive Inhalte: mit CSS und JavaScript umsetzbar
- Flash und Sicherheitsprobleme
  - iOS unterstützt Flash gar nicht
  - Browserhersteller planen, zukünftig Flash standardmäßig zu deaktivieren



openHPI: **Web-Technologien**

DOM – Document Object Model

Prof. Dr. Christoph Meinel

Hasso-Plattner-Institut

Universität Potsdam

DOM – Document Object Model

Wie kann mit einer Programmiersprache auf Hypermedia-Dokumente und deren Inhalt zugegriffen werden?

Document Object Model – DOM

- Grundlage für Auslesen aus/Manipulieren von bestehenden HTML-Dokumenten auf Client-Seite
- Ursprünglich plattform- & sprachunabhängige Schnittstelle zum Zugriff auf Dokumente(teile)
- Vom W3C standardisiert
  - bildet HTML-Baumstruktur in Klassen und Objekte ab
  - zur Navigation und Manipulation der Bäume
  - Ereignismodell
  - Unterstützung von XML-Namespaces

DOM: Objekt-Modell (1/6)

Aufbau

- DOM modelliert Dokument entsprechend seiner Struktur als Strukturbaum
- DOM definiert Objekte mit zugehörigen Attributen und Methoden zur Manipulation der Objekte
- DOM legt fest:
  - Schnittstellen und Objekte zur Darstellung/Manipulation von Dokumenten
  - Semantik der definierten Objekte/Schnittstellen zur Beschreibung von Attributen und Verhalten
  - Beziehungen und Kompatibilität der einzelnen Objekte/Schnittstellen

DOM: Objekt-Modell (2/6)

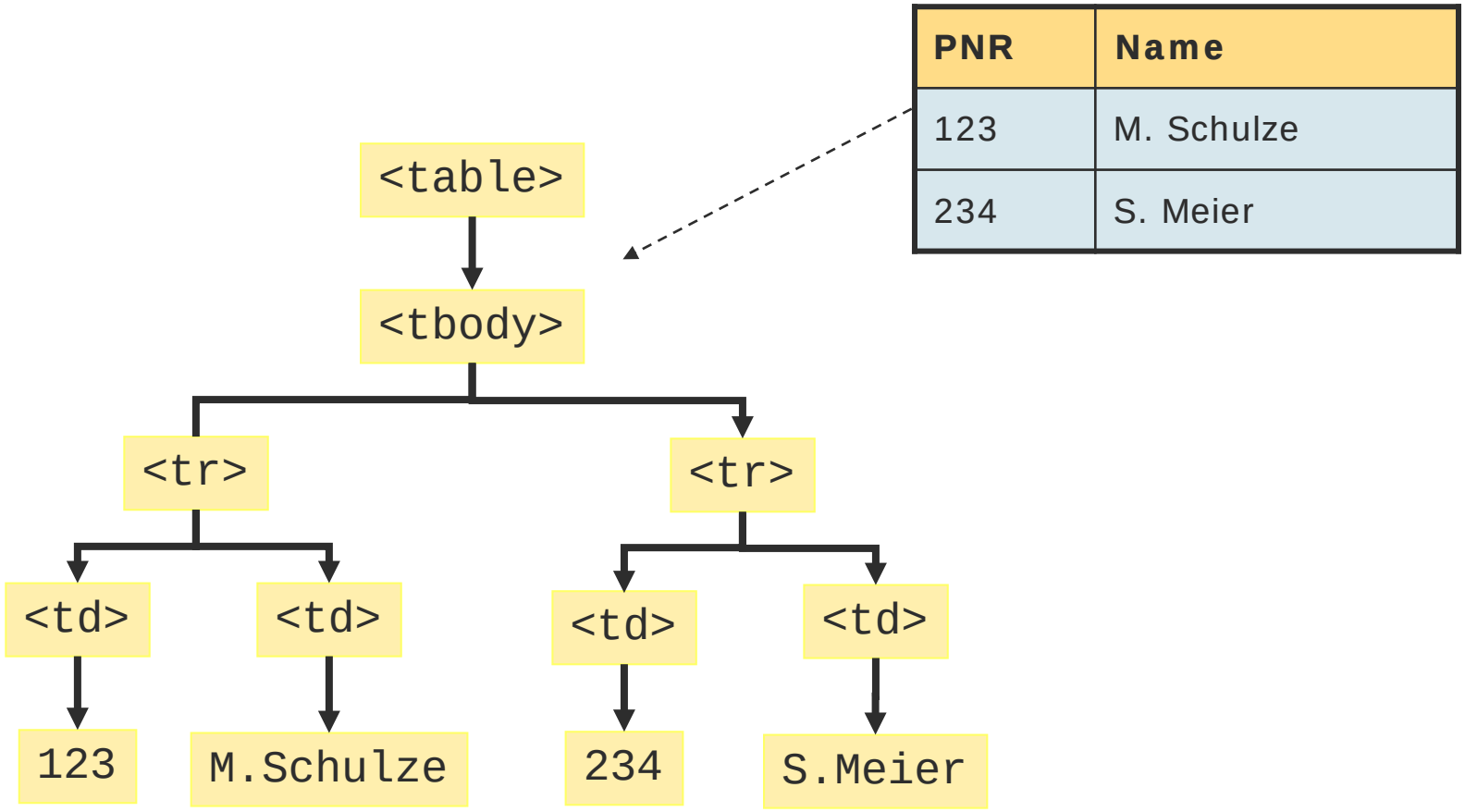
HTML-Code und seine Darstellung

```
<table>
…
  <tbody>
    <tr>
      <td>123</td>
      <td>M. Schulze</td>
    </tr>
    <tr>
      <td>234</td>
      <td>S. Meier</td>
    </tr>
  </tbody>
</table>
```

PNR	Name
123	M. Schulze
234	S. Meier

DOM: Objekt-Modell (3/6)

Darstellung als Strukturbaum



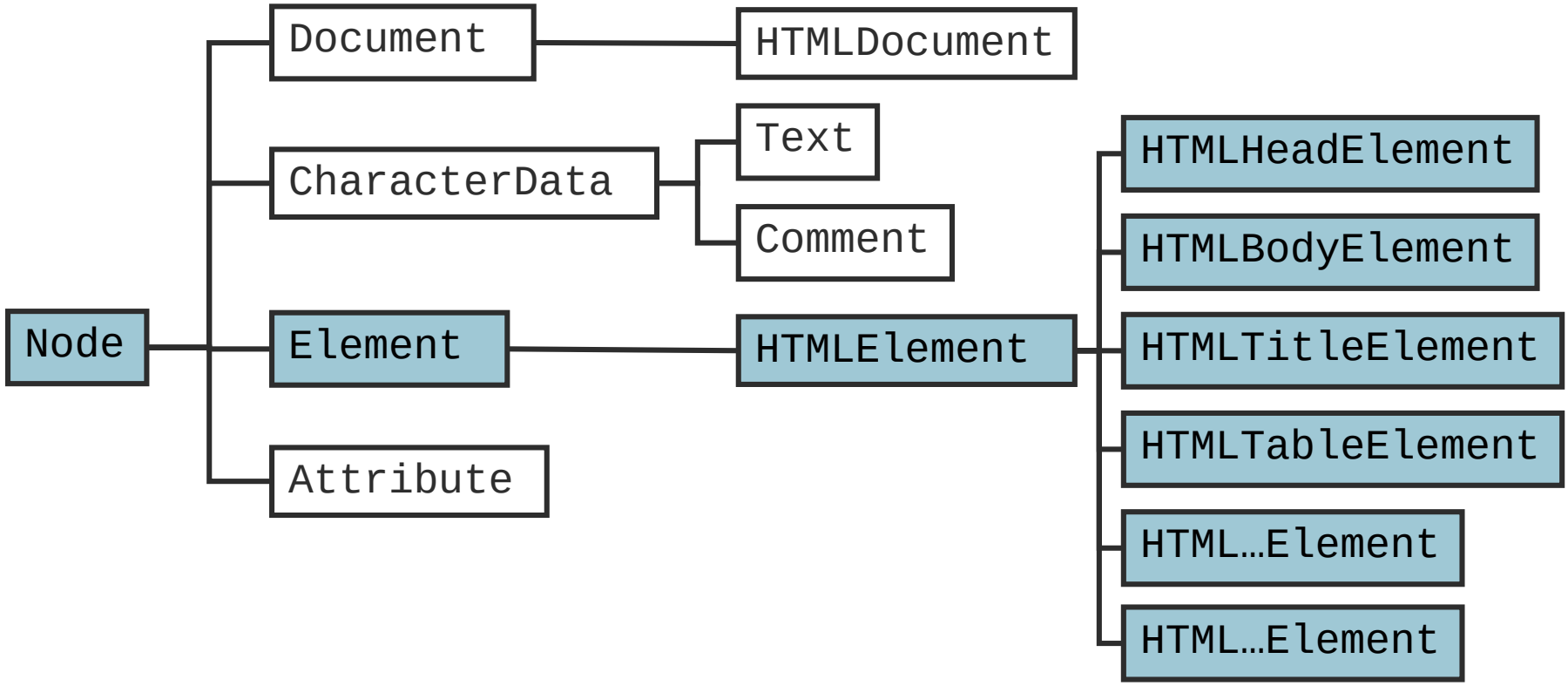
DOM: Objekt-Modell (4/6)

Klassenhierarchie

- Zu jedem Dokument gehören:
  - Wurzelknoten
  - Elementknoten
  - Attributknoten
  - Textknoten
  - (Kommentarknoten)
- DOM ist eine Spezifikation, definiert Schnittstellen
- Programmiersprachen (v.a. JavaScript) implementieren diese
- Implementierungen in anderen Sprachen vorhanden, z.B.
  - für Java: Dom4J, JDOM, ...

DOM: Objekt-Modell (5/6)

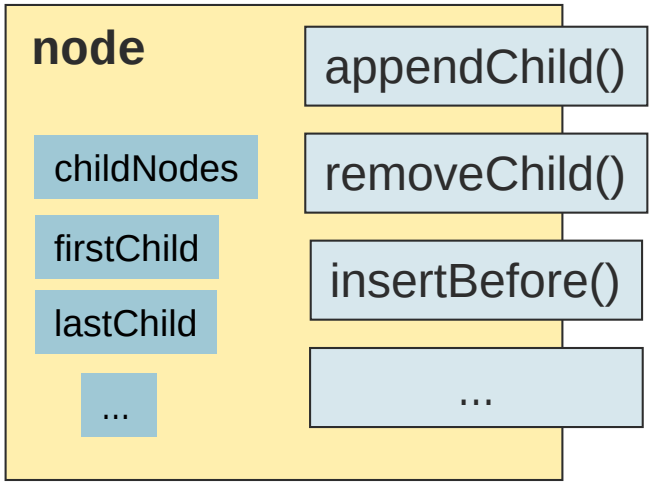
Klassenhierarchie



DOM: Objekt-Modell (6/6)

Zentrale Klasse: Node

- Attribute:
  - node.childNodes
  - node.firstChild
  - node.lastChild
  - node.parentNode
  - ...
- Methoden:
  - node.appendChild()
  - node.removeChild()
  - node.insertBefore()
  - node.cloneNode()
  - ...



openHPI: **Web-Technologien**

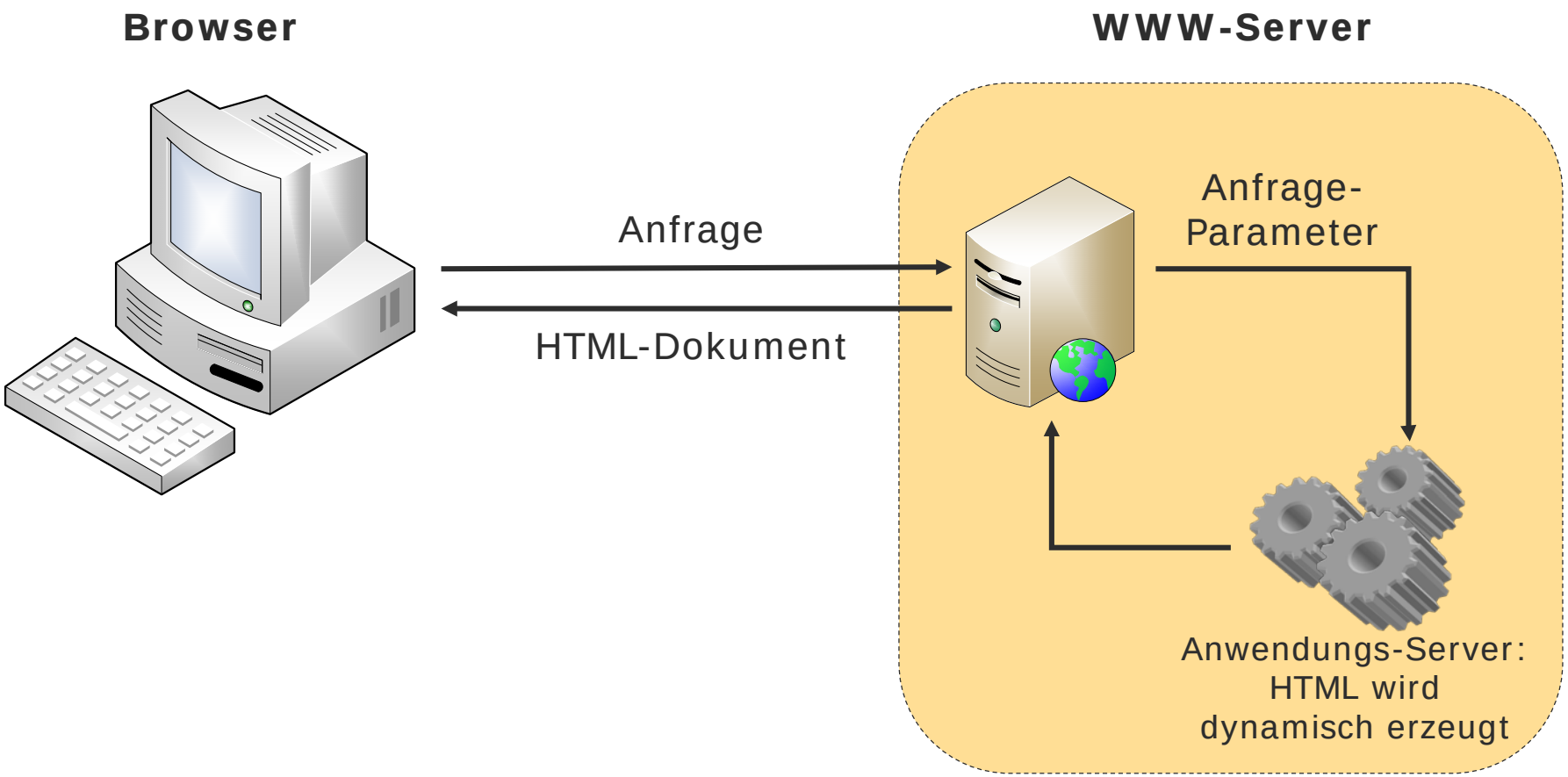
Serverseitige Web-Programmierung

Prof. Dr. Christoph Meinel

Hasso-Plattner-Institut

Universität Potsdam

Dynamische Webseiten



Anwendungsserver

- Statt statische HTML-Dateien auszuliefern, können HTTP-Server Erstellung der Website **Anwendungsprogrammen** auf dem Server überlassen, die dynamisch HTML erzeugen und dieses an WWW-Client ausliefern
- Standardisierte Schnittstelle zwischen HTTP-Server und Anwendungsprogrammen auf Serverseite:
  Common Gateway Interface (CGI)
- Anwendungsprogramme können mit beliebigen Programmiersprachen erstellt werden, z.B.
  - Java (Servlets, Server Pages, Beans, ...)
  - Skriptsprachen (ASP, PHP, Python, Ruby, JavaScript, ...)
- **Web-Frameworks**, z.B. Ruby on Rails, Spring, Django, ... liefern Grundgerüst und erleichtern Umsetzung üblicher Komponenten

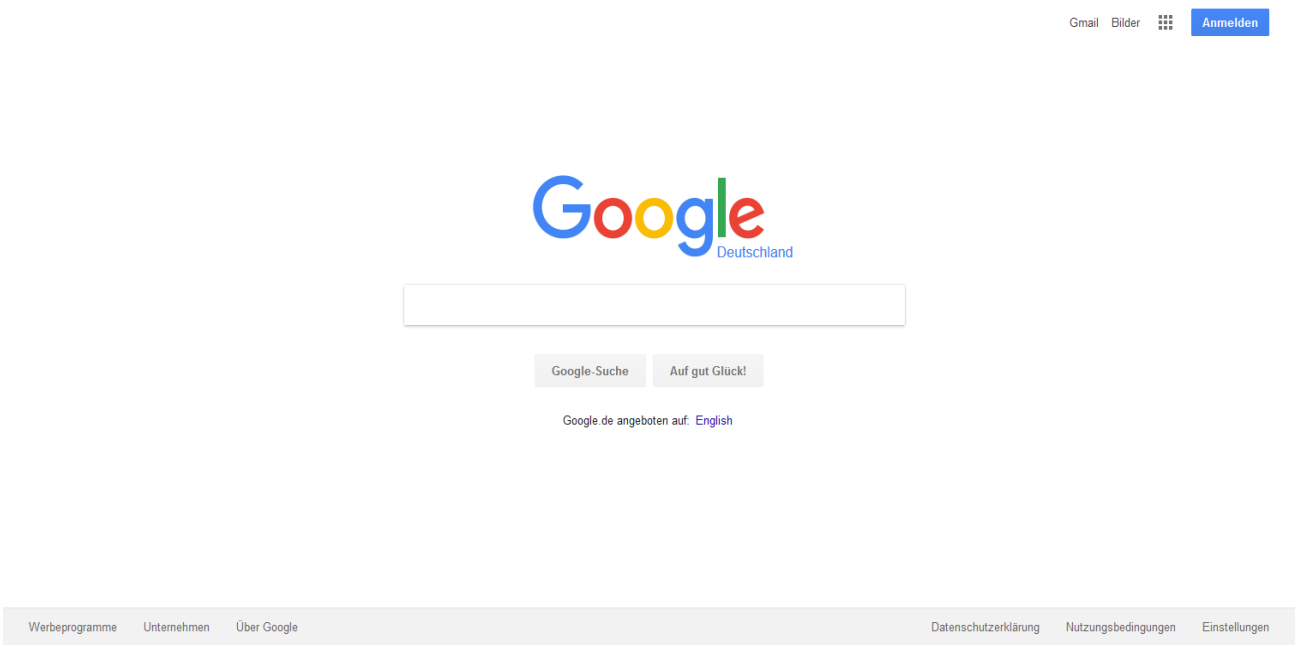
Serverseitige Skriptsprachen

Voraussetzung: Server verfügt über entsprechendes Modul/Interpreter

- Zwei Alternativen
  - Über spezielle Tags wird Code direkt in HTML-Dateien eingebunden
    - Beispiel: PHP: <?php echo "Hello PHP!" ?>
    - Fehlt Modul auf dem Server, wird Quellcode nicht ausgeführt, sondern angezeigt
  - Applikations-Server implementiert selbst HTTP-Schnittstelle
- **Vorteil**
  - Schnelle Entwicklung durch direkte Manipulation der HTML-/CSS-/JS-Elemente
- **Nachteile**
  - oft Vermischung von Darstellung und Applikationslogik
  - Performance: Code muss interpretiert werden

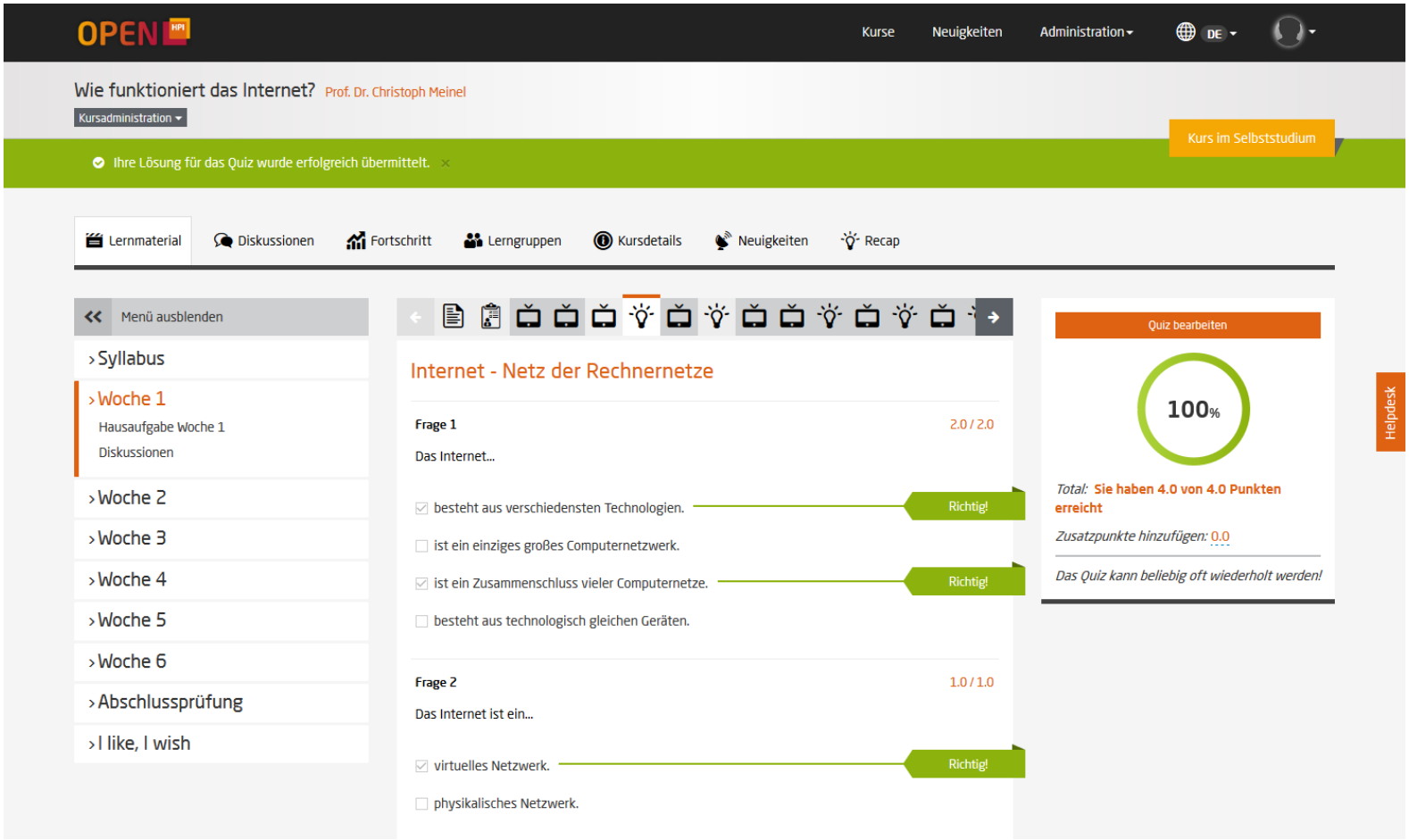
Anwendungsbeispiele (1/3)

Suchmaschinen



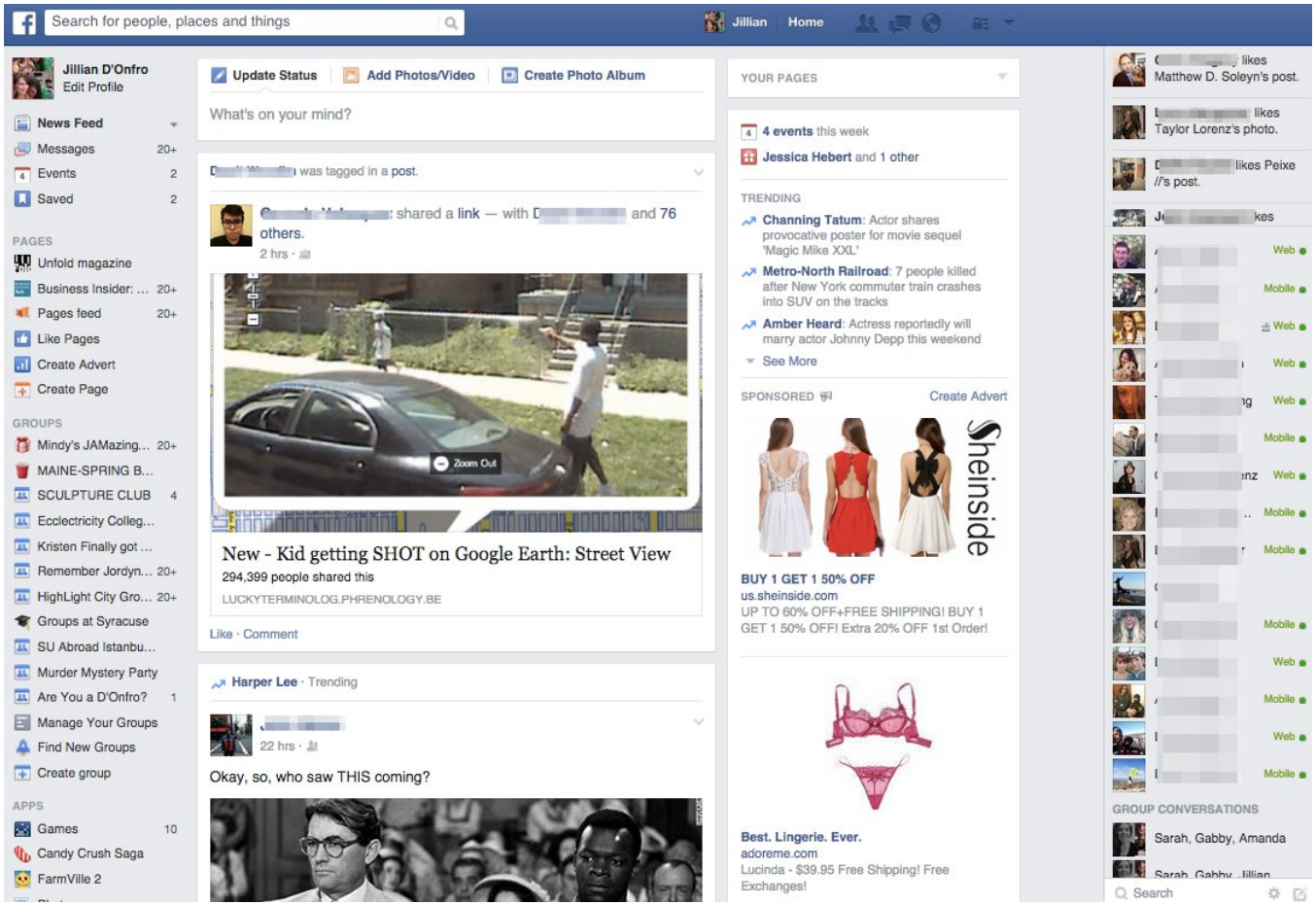
Anwendungsbeispiele (2/3)

Online-Kurse



Anwendungsbeispiele (3/3)

Soziale Netzwerke



openHPI: **Web-Technologien**

Session-Management

Prof. Dr. Christoph Meinel

Hasso-Plattner-Institut

Universität Potsdam

Session-Management

Viele Web-Anwendungen müssen sich an frühere Interaktionen zwischen Browser und WWW-Server erinnern „können“, z.B.

- virtuelle Warenkörbe beim Online-Shopping,
- Webseiten mit Login, usw.

Aber HTTP ist **zustandsloses Protokoll**. Deshalb braucht es Konzepte zur Bildung von „**Sessions“**:

- Zusammenfassung zusammengehörenden Nutzerinteraktionen (Request-Response-Zyklen) zwischen Client und Server
- Sessions werden vermittels einer **Session-ID** identifiziert, die übertragen werden können durch
  - **Cookies** – Key-Value-Paare im HTTP-Header (→ Woche 1)
  - „**Hidden Fields**“ in HTML-Formularen
  - **URL Rewriting**

Session-Management: URL Rewriting

URL Rewriting ist sehr simple, aber unzureichende Methode der Zustandslosigkeit von HTTP zu begegnen

- Beispiel `http://example.org/index.html?sessionid=1234567890`

Vorteile

- Einfach zu realisieren
- Session-Informationen für Nutzer einfach in der URL ablesbar

Nachteile

- Länge von URLs ist begrenzt
- da bei unverschlüsselter Verbindung die URL einfach mitgelesen werden kann, ist URL Rewriting nicht geeignet für sensitive Daten, z.B.: `http://example.org/login/?username=openhpi&password=12345`
- Session kann bei Kenntnis der URL durch Dritte einfach übernommen (gekapert) werden

Session-Management: Hidden Fields

Hidden Fields sind versteckte Felder in HTML-Formularen

- Formular besitzt Felder vom Typ **hidden**, für einen Namen und einen Wert, z.B. Session-ID
- Abgeschickt werden diese Formulare mit HTTP POST-Methode

```
<form name="session" method="post"
      action="dashboard.html">
  <input type="hidden"
         name="SessionID"
         value="1234567890" />

  <input type="submit"
         value="Zum Dashboard">
</form>
```

Vorteil

- Bei Übertragung mit HTTPS bleiben Session-Informationen geheim

Nachteile

- Session-Informationen sind im Quelltext der Seite auslesbar, wenn Seite unverschlüsselt übertragen wird
- Einfache Links so nicht möglich, da gesamte Navigation über Formulare realisiert werden muss

Session-Management: Cookies (1/2)

Cookie-Mechanismus (Erinnerung Woche 1):

1. HTTP-Server beauftragt Client, Cookie anzulegen, indem er im HTTP response das Headerfeld **`set-cookie`** setzt
2. Browser speichert Cookie in internen Datenbank
3. Bei jedem neuen Request an gleiche Domain sendet Browser all seine dazu passenden Cookies im Request-Headerfield **`cookie`**

- Typischerweise werden nur (Session) IDs in Cookies gesendet, eigentliche Informationen (z.B. virtueller Warenkorb) werden serverseitig abgelegt

Session-Management: Cookies (2/2)

Cookie-Mechanismus (Erinnerung Woche 1):

Vorteile

- Bei Übertragung mit HTTPS bleiben Informationen geheim
- Einfache Wartbarkeit für Webseitenbetreiber und –Entwickler
- Session-Handling unabhängig von HTML

Nachteil

- Session-Informationen aus HTTP-Header auslesbar, wenn diese unverschlüsselt übertragen wird
- Tracking von Nutzer-Interaktionen sehr einfach möglich für Webseitenbetreiber und Dritte (Google, Facebook, …)

openHPI: **Web-Technologien**

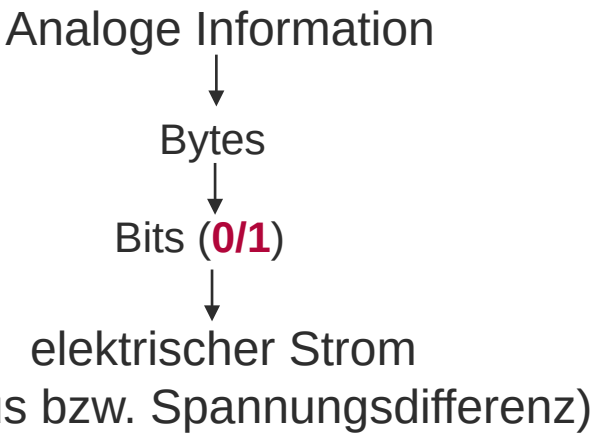
Informationskodierung

Prof. Dr. Christoph Meinel
Hasso-Plattner-Institut
Universität Potsdam

Kodierung von Information (1/3)

„Natürliche Information“ ist **analog**, z.B. **Musik / Audioinformation**:

- Töne sind charakterisiert durch Tonhöhe, Klang, Lautstärke, ...
  aber Darstellung im Computer ist **digital**, d.h.
- Informationen müssen **kodiert** werden, um sie
  - im Computer zu verarbeiten und
  - über das Internet übertragen zu können



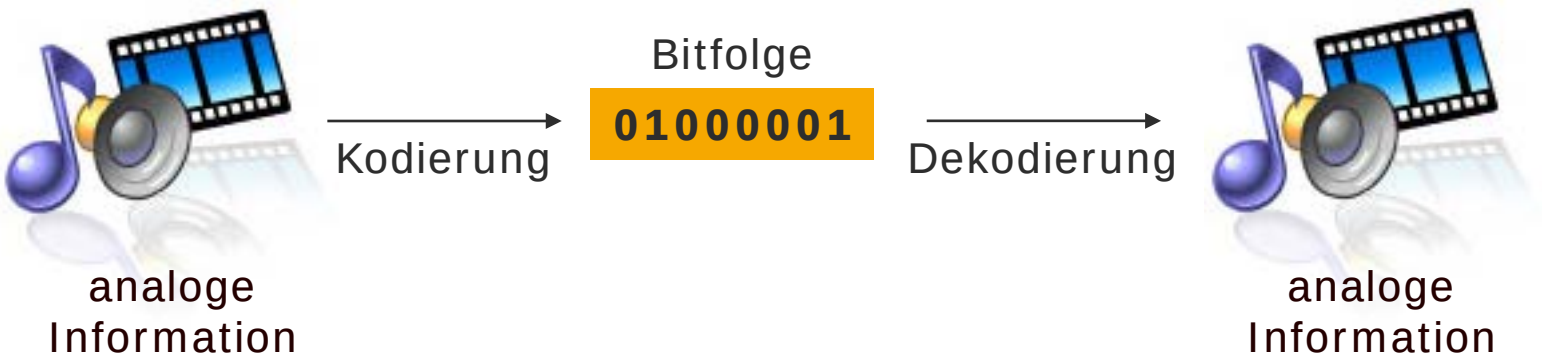
Kodierung von Information (2/3)

Kodierung

Vorgang der Umwandlung einer Information

- aus einer bestimmten Darstellung (Ausprägung)
- in eine andere Darstellung ohne (bzw. mit zu tolerierendem) Informationsverlust

Rückwandlung wird als **Dekodierung** bezeichnet



Kodierung von Information (3/3)

Zeitunabhängige Medien

- Zeitkomponente bei Aufzeichnung und Wiedergabe ohne Bedeutung, z.B. Text, Grafik

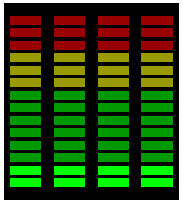
Zeitabhängige Medien

- Information verändert sich mit der Zeit, z.B. Audio, Video
- Gehalt von Einzelinformation ist dabei oft nicht so signifikant
- Gesamtinformation erschließt sich erst aus zeitlichem Ablauf

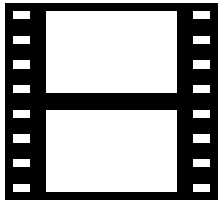
ABC 123

alphanumerische Information

grafische Information



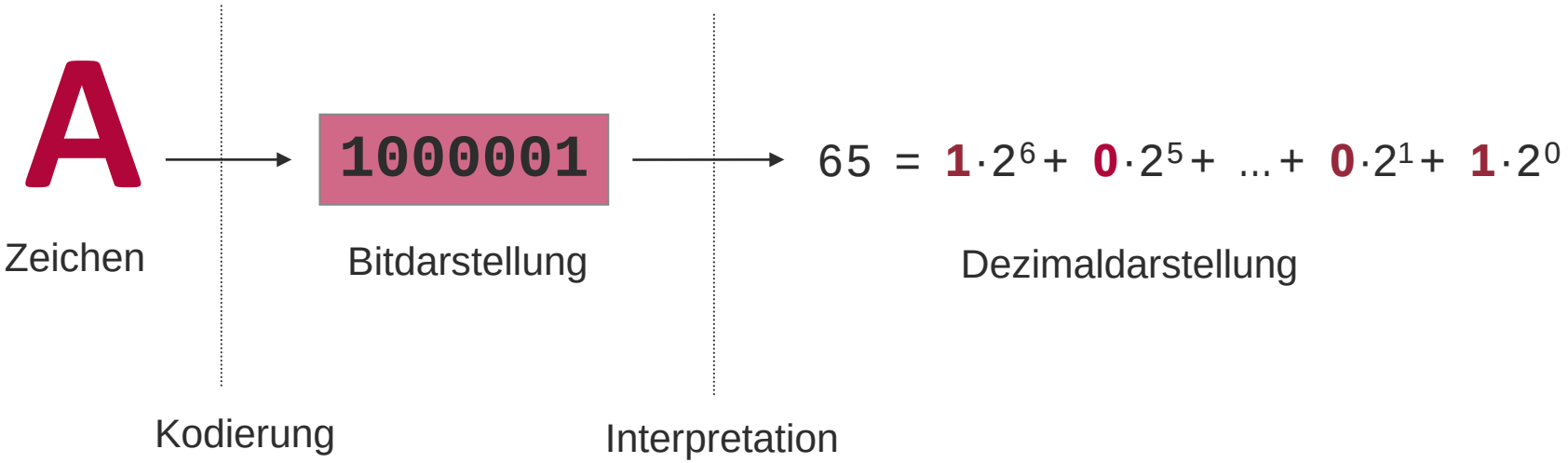
Audio-Information



Video-Information

Kodierung von Zeichen (1/4)

Zeichen müssen in ein Format umgewandelt werden, mit dem Computer umgehen können



Einfaches Beispiel:

ASCII – American Standard Code for Information Interchange

ASCII – 7-Bit-Kodierung mit 128 Zeichen:

ASCII value	Character	Control character	ASCII value	Character	ASCII value	Character	ASCII value	Character
000	(null)	NUL	032	(space)	064	@	096	
001	☺	SOH	033	!	065	A	097	a
002	☻	STX	034	"	066	B	098	b
003	♥	ETX	035	#	067	C	099	c
004	♦	EOT	036	$	068	D	100	d
005	♣	ENQ	037	%	069	E	101	e
006	♠	ACK	038	&	070	F	102	f
007	(beep)	BEL	039	'	071	G	103	g
008	◘	BS	040	(	072	H	104	h
009	(tab)	HT	041	)	073	I	105	i
010	(line feed)	LF	042	*	074	J	106	j
011	(home)	VT	043	+	075	K	107	k
012	(form feed)	FF	044	,	076	L	108	l
013	(carriage return)	CR	045	-	077	M	109	m
014	♫	SO	046	.	078	N	110	n
015	☼	SI	047	/	079	O	111	o
016	►	DLE	048	0	080	P	112	p
017	◄	DC1	049	1	081	Q	113	q
018	↕	DC2	050	2	082	R	114	r
019	!!	DC3	051	3	083	S	115	s
020	¶	DC4	052	4	084	T	116	t
021	§	NAK	053	5	085	U	117	u
022	▬	SYN	054	6	086	V	118	v
023	↨	ETB	055	7	087	W	119	w
024	↑	CAN	056	8	088	X	120	x
025	↓	EM	057	9	089	Y	121	y
026	→	SUB	058	:	090	Z	122	z
027	←	ESC	059	;	091	[	123	{
028	(cursor right)	FS	060	<	092	\	124	|
029	(cursor left)	GS	061	=	093	]	125	}
030	(cursor up)	RS	062	>	094	^	126	~
031	(cursor down)	US	063	?	095	_	127	⌂

Kodierung von Zeichen (3/4)

Mit 7-Bit-ASCII lassen sich $2^7 = 128$ verschiedene Zeichen kodieren

Um mehr Zeichen kodieren zu können, sind mehr Bits notwendig:

- **ISO 8859**: 8-Bit-Kodierung – 256 Zeichen
  - 0-127 identisch mit ASCII
  - 128-159 seltene Steuerzeichen
  - 160-255 (multi-)nationale Erweiterungen wie ¿ ¡ ñ ç ä ö ü ß ญ
    ...

ISO 8859-1	Westeuropäisch
ISO 8859-5	Kyrillisch
ISO 8859-11	Thailändisch
...	

Reicht aber noch nicht für **Chinesisch**, wo im Alltag etwa 4000 Zeichen benötigt werden (insgesamt ca. 87.000 Zeichen) ...

Kodierung von Zeichen (3/3)

Reicht aber noch nicht für **Chinesisch**, wo im Alltag etwa 4000 Zeichen benötigt werden (insgesamt ca. 87.000 Zeichen) …

Unicode – UTF (Universal Transformation Formats)

- UTF-8: Unicode mit mind. 8 Bit pro Zeichen
  - kompatibel mit ASCII
  - chinesische Zeichen werden mit 3 oder 4 UTF-8-Blöcken kodiert
- UTF-16: Unicode mit 16 oder 32 Bit pro Zeichen (2 bzw. 4 Byte)
- UTF-32: Unicode mit 32 Bit pro Zeichen (4 Byte)

Kodierung und Übertragung von multimedialen Daten

Im Web müssen viele Daten übertragen werden, typischerweise mehr Daten vom Server zum Client als andersherum, z.B. YouTube

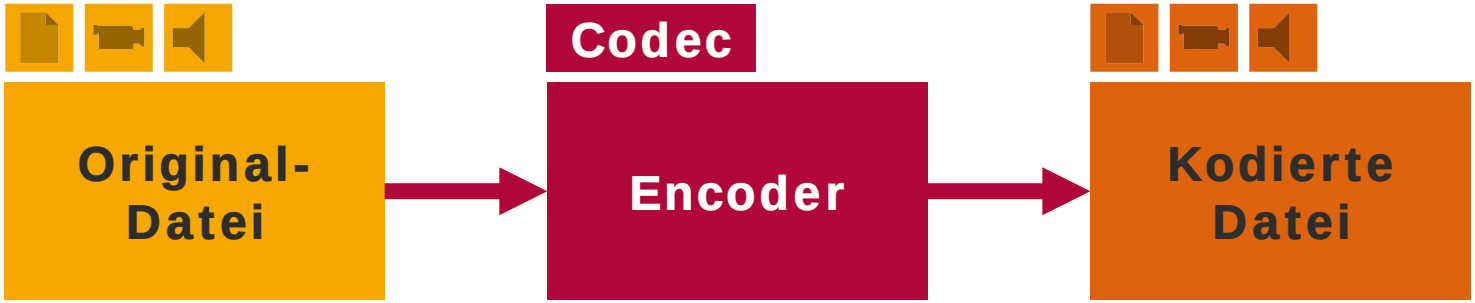
- Um Daten versenden und empfangen zu können, müssen sie zunächst kodiert werden
- Kodierungsverfahren (der **Codec**) müssen standardisiert sein, sonst sind Webbrowser nicht in der Lage, empfangene Daten korrekt zu „verstehen“, d.h. Text/Bilder/Videos-Daten richtig zu dekodieren und darzustellen
- Verringerung der Datenmenge durch **Komprimierung**
  - **Verlustfreie** Komprimierung: Verringerung der Redundanz einer Datenmenge
  - **Verlustbehaftete** Komprimierung: Weglassen von Information, die „nicht so wichtig“

Komprimierung von Dateien

Immer mehr multimediale-Daten → wachsender Bedarf an Speicherplatz und Übertragungsbandbreite

- Bilder, Audio und Video verbrauchen viel mehr Speicher als Text
- Müssen versuchen, Speicherbedarf zu minimieren

→ **Komprimierung** der Daten in ein standardisiertes Format, ggf. unter Inkaufnahme von vertretbarem **Informationsverlust**



→ Schauen uns das in den nächsten Einheiten genauer an für Grafik-, Audio- und Video-Daten an

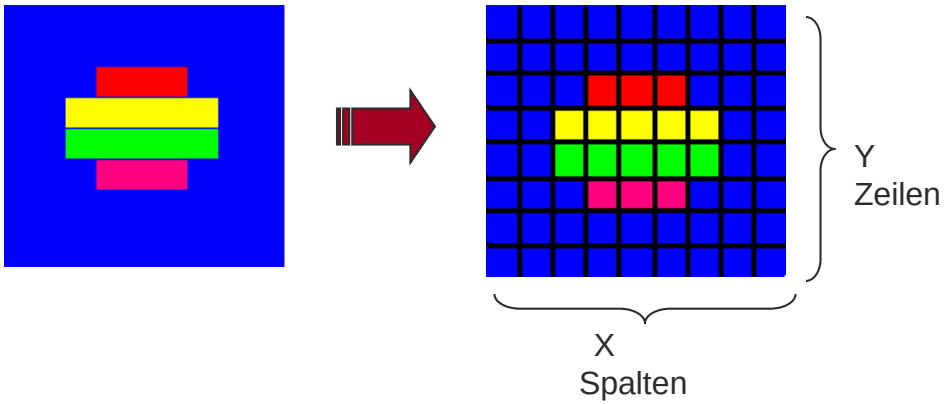
openHPI: **Web-Technologien**
Grafikkodierung

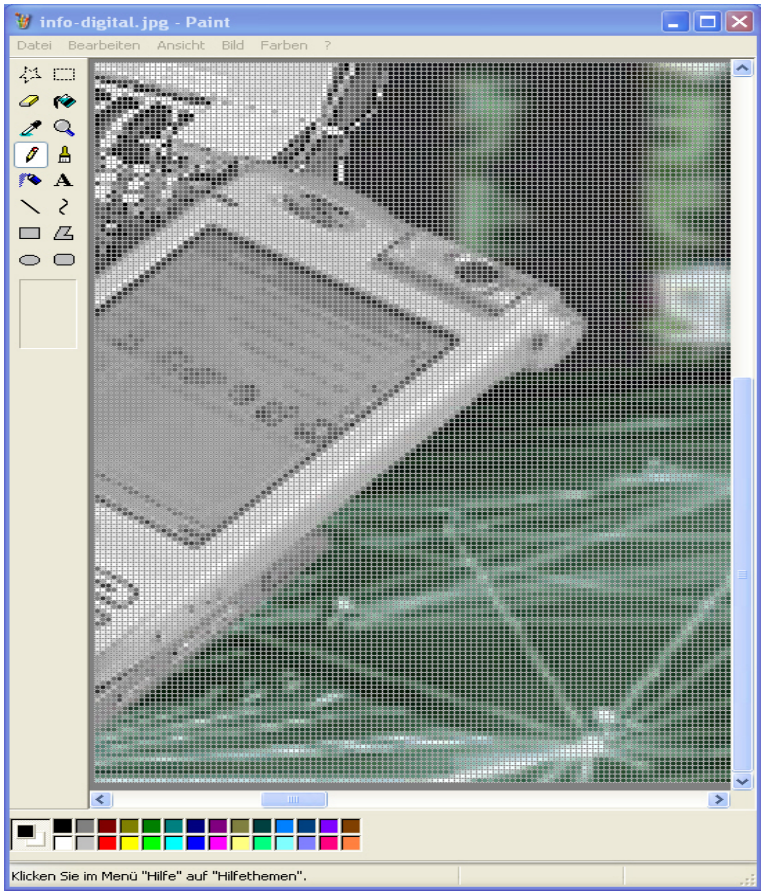
Prof. Dr. Christoph Meinel
Hasso-Plattner-Institut
Universität Potsdam

Grafikkodierung: Bitmap-Grafik (1/3)

Bitmap-Grafik

- Grafik wird aufgerastert in Matrix aus einzelnen Bildpunkten (Pixel) „Rastergrafik“
- Jeder Bildpunkt ist charakterisiert durch
  - Helligkeit und
  - Farbe

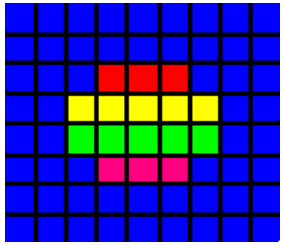




Grafikkodierung: Bitmap-Grafik (2/3)

Bitmap-Grafik

- Entstanden zusammen mit Entwicklung der **Kathodenstrahlröhre**
  - Jegliche Information (Farbe, Helligkeit) wird für jeden einzelnen Pixel gespeichert
  - Hoher Speicherplatzbedarf, daher nicht geeignet für das Web



- Besser: Komprimierte Bitmap-Formate, die z.T. (unnötige) Daten weglassen um Speicherplatz einsparen, z.B. **JPEG**, **PNG**, **TIFF**, ...

Beispiel:

BMP, 88 KB

JPEG, 27 KB

JPEG, 2 KB

Grafikkodierung: Bitmap-Grafik (3/3)

Unkomprimierte Bitmaps

- Jedes Pixel wird einzeln gespeichert, hoher Speicherplatzbedarf
- Typisches Dateiformat: **BMP**

Komprimierte Bitmaps

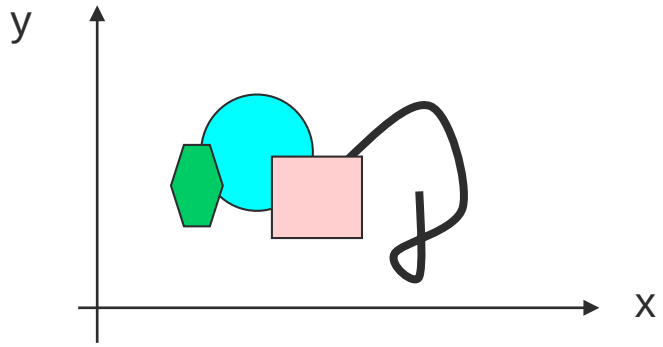
- Unnötige Information vermeiden, z.B.
  - einzelne Pixel zu Linien oder Bereichen gleicher Farbe/Helligkeit zusammenfassen
- Einige Verfahren sind verlustbehaftet (**JPEG**); andere können je nach Einstellung verlustfrei benutzt werden (**PNG**, **TIFF**, **GIF**)

Vektorgrafik

- (Künstliche) Grafiken werden durch Vektoren beschrieben (SVG)

Vektorgrafik

- Mathematisches Format, das auf Beschreibung geometrischer Primitive basiert, z.B. Punkte, Linien, Flächen, Kurven, Polygone, ...
- Objekte werden beschrieben durch Vektoren mit Startpunkt (x,y) und Richtung; Farben, Linienstärke werden als Attributinformation hinzugefügt
- Programm rekonstruiert daraus darzustellende Figur



- Entstanden zusammen mit Entwicklung des **Plotters**

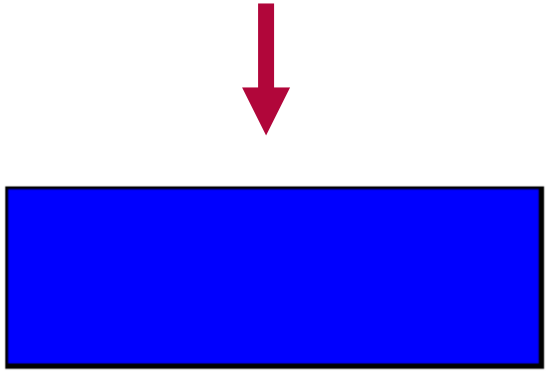
Grafikkodierung: Vektorgrafik (2/4)

Beliebtestes Format: **SVG – Scalable Vector Graphics**

- **Vorteile**:
  - **skalierbar**, also beliebig weit zoombar ohne Treppen-/Pixeleffekte
  - niedriger Speicherbedarf
  - XML-basiertes Format (nachträgliche Transformationen mit XML-Mitteln möglich)
- **Nachteil**:
  - nur für “künstliche” Grafiken geeignet, nicht für Fotos etc.)

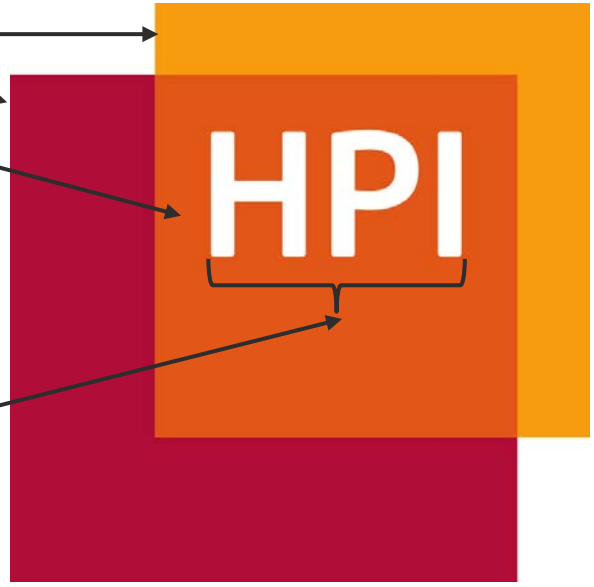
SVG – Einfaches Beispiel:

```
<svg width="400" height="110">
  <rect width="300" height="100“
    style="fill:rgb(0,0,255);
    stroke-width:3;stroke:rgb(0,0,0)" />
</svg>
```



SVG – Etwas komplexeres Beispiel:

```
<svg xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns="http://www.w3.org/2000/svg" height="141.73" width="141.73"
version="1.1" xmlns:cc="http://creativecommons.org/ns#"
xmlns:dc="http://purl.org/dc/elements/1.1/">
<g transform="translate(-222.01 -445.84)">
<path style="fill:#b00c36" d="m222.01 587.57h124.01v-124.02h-124.01v124.02z"/>
<path style="fill:#f89c0e" d="m257.44 552.14h106.3v-106.3h-106.3v106.3z"/>
<path style="fill:#e25617" d="m257.44 552.14h88.58v-88.582h-88.58v88.582z"/>
<path style="fill:#ffffff" d="m293.47 477.31h-4.0835c-0.45413 0-0.82525 0.33-
0.82525 0.78375v11.386h-11.715v-11.386c0-0.45375-0.371-0.78375-0.82512-
0.78375h-4.0835c-0.45412 0-0.82525 0.33-0.82525 0.78375v29.04c0 0.45375
0.37113 0.825 0.82525 0.825h4.0835c0.45412 0 0.82512-0.37125 0.82512-0.825v-
12.622h11.715v12.622c0 0.45375 0.37112 0.825 0.82525 0.825h4.0835c0.454 0
0.82512-0.37125 0.82512-0.825v-29.04c0-0.45375-0.37112-0.78375-0.82512-
0.78375zm16.488-0.33c-3.0118 0-5.94 0.20625-8.086 0.53625-0.82512 0.12375-
1.3192 0.41375-1.3192 1.2375v28.38c0 0.45375 0.37112 0.825 0.82512
0.825h4.1245c0.45413 0 0.78375-0.37125 0.78375-0.825v-10.766c0.94875 0.0412
2.5991 0.165 3.6719 0.165 6.3525 0 11.88-2.3512 11.88-9.6525v-0.37125c0-
7.3012-5.5274-9.5288-11.88-9.5288zm6.1865 9.9c0 3.3-1.7319 4.9512-6.1865
4.9512-0.70163 0-3.0118-0.0838-3.6719-0.125v-9.8175c0.74213-0.0825 2.8051-
0.165 3.6719-0.165 4.125 0 6.1865 1.32 6.1865 4.785v0.37125zm15.378-9.57h-
4.0845c-0.45413 0-0.82525 0.33-0.82525 0.78375v29.04c0 0.45375 0.37113 0.825
0.82525 0.825h4.0845c0.45312 0 0.82512-0.37125 0.82512-0.825v-29.04c0-0.45375-
0.372-0.78375-0.82512-0.78375"/>
</g>
</svg>
```



Grafikkomprimierung mit JPEG

Originalbild

komprimiertes Bild

Farbraum-wechsel

Chroma-Subsampling

Diskrete Cosinus-Transformation

Quantisierung

modifizierte Huffman-Kodierung

openHPI: **Web-Technologien**

Audiokodierung

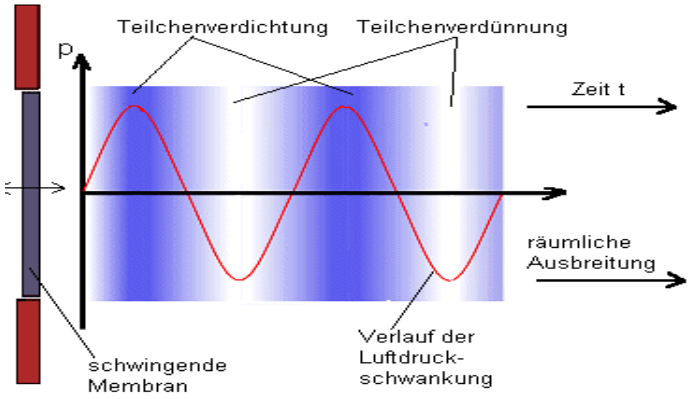
Prof. Dr. Christoph Meinel
Hasso-Plattner-Institut
Universität Potsdam

Audiokodierung und –komprimierung: Schall und seine Digitalisierung

Schall ist analoges Signal

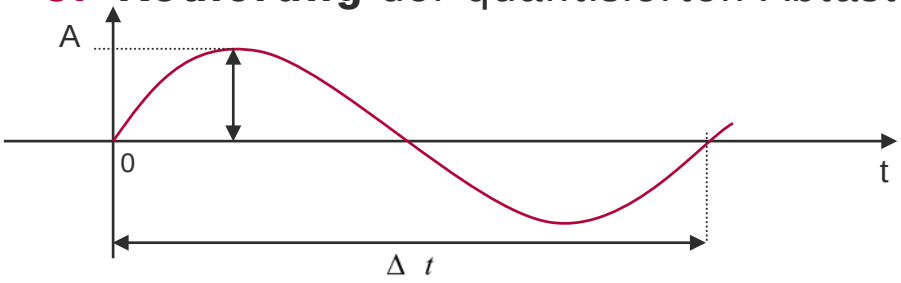
- zeitkontinuierlich
- wertekontinuierlich

Vor Darstellung im Computer und Übertragung im Web müssen akustische Signale digitalisiert werden



Analog-Digital-Wandlung erfolgt in drei Stufen:

1. **Sampling** – Abtastung des Signals
2. **Quantisierung** – Diskretisierung der Abtastwerte
3. **Kodierung** der quantisierten Abtastwerte

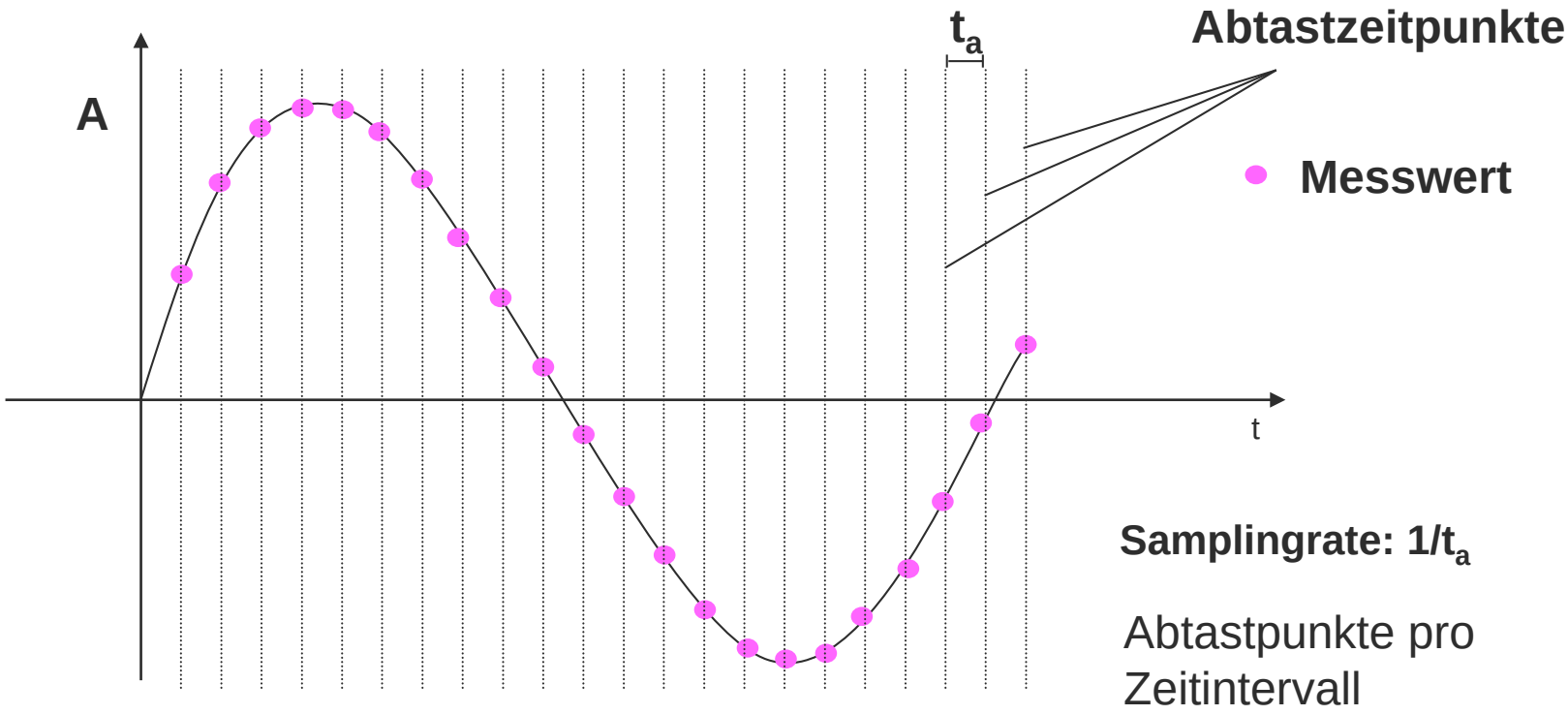


0110010010010011 0.....

Audiokodierung und –komprimierung: Sampling und Quantisierung (1/6)

Sampling:

Signal wird periodisch in bestimmten Zeitabständen t_a abgetastet



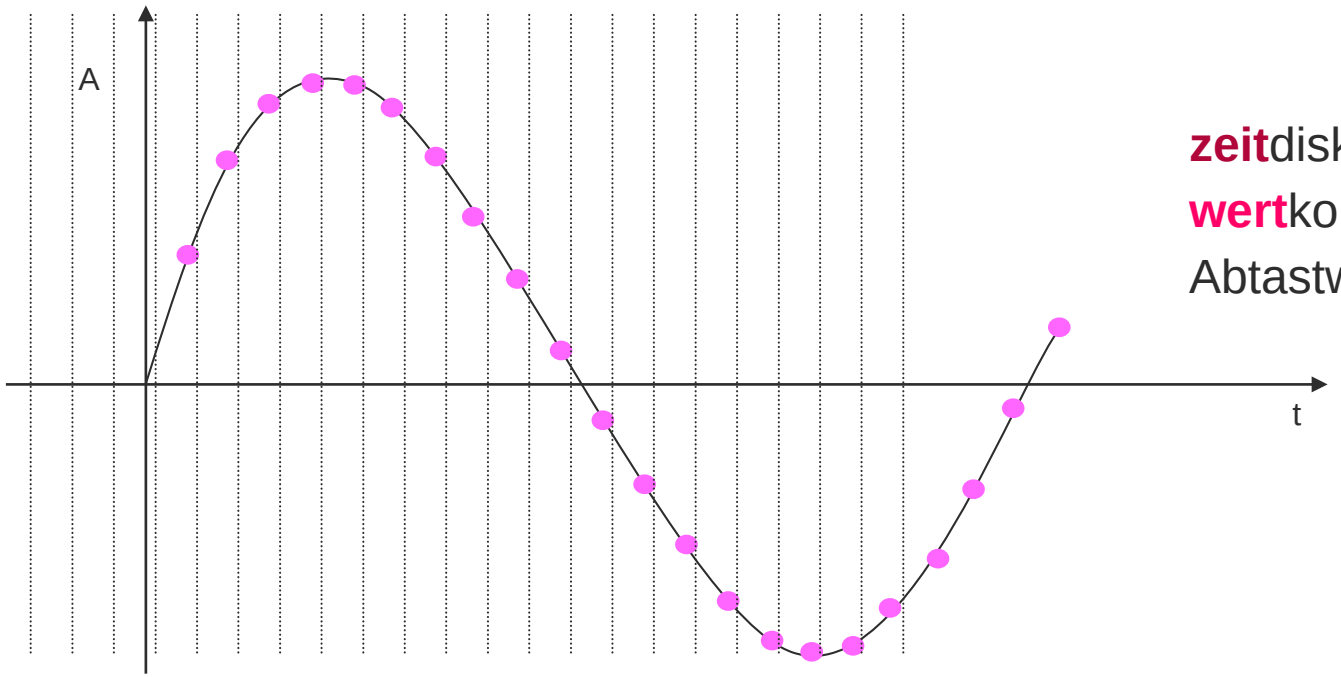
Samplingrate: $1/t_a$

Abtastpunkte pro Zeitintervall

Audiokodierung und –komprimierung: Sampling und Quantisierung (2/6)

Sampling:

Signal wird periodisch in bestimmten Zeitabständen t_a abgetastet

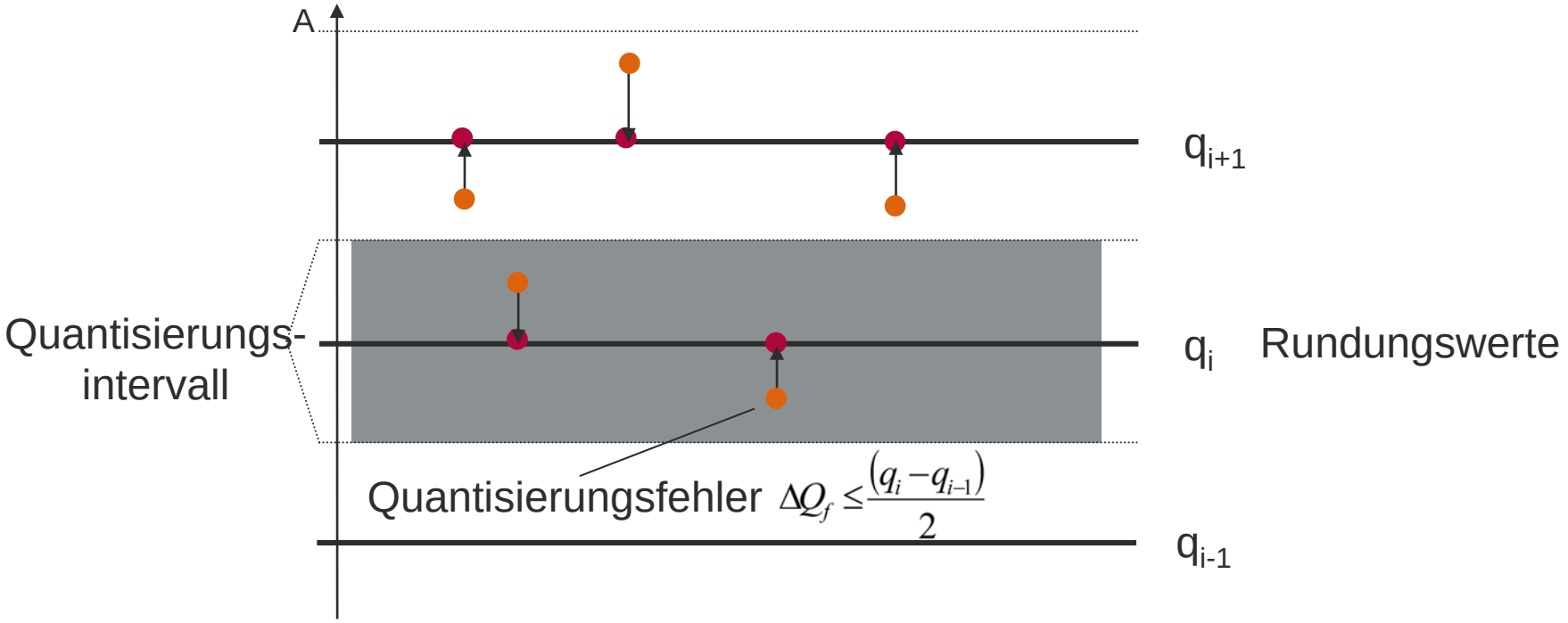


zeitdiskrete, aber
wertkontinuierliche
Abtastwerte

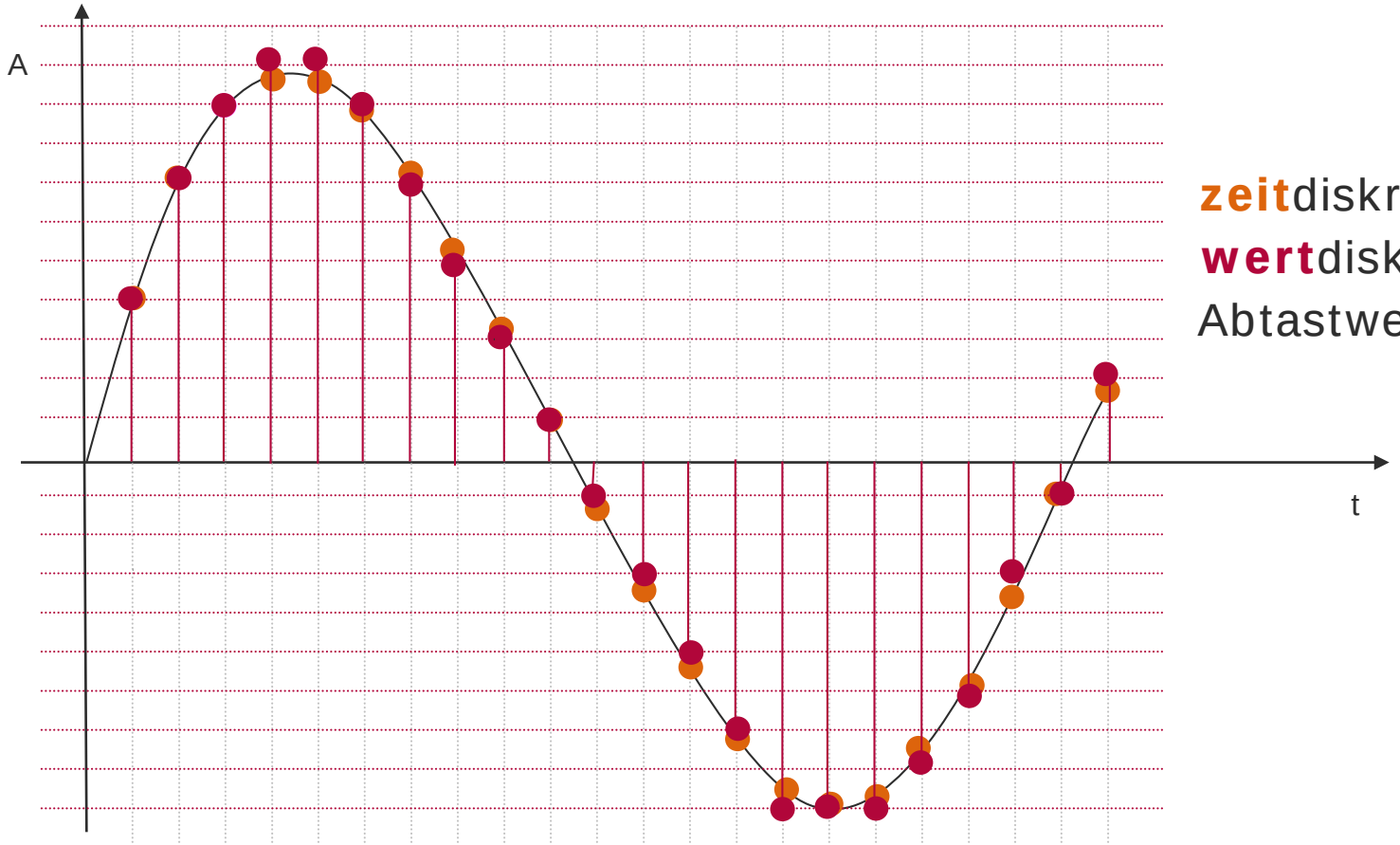
Audiokodierung und –komprimierung: Sampling und Quantisierung (3/6)

Quantisierung:

Rundung der kontinuierlichen Abtastwerte auf diskrete Quantisierungspunkte

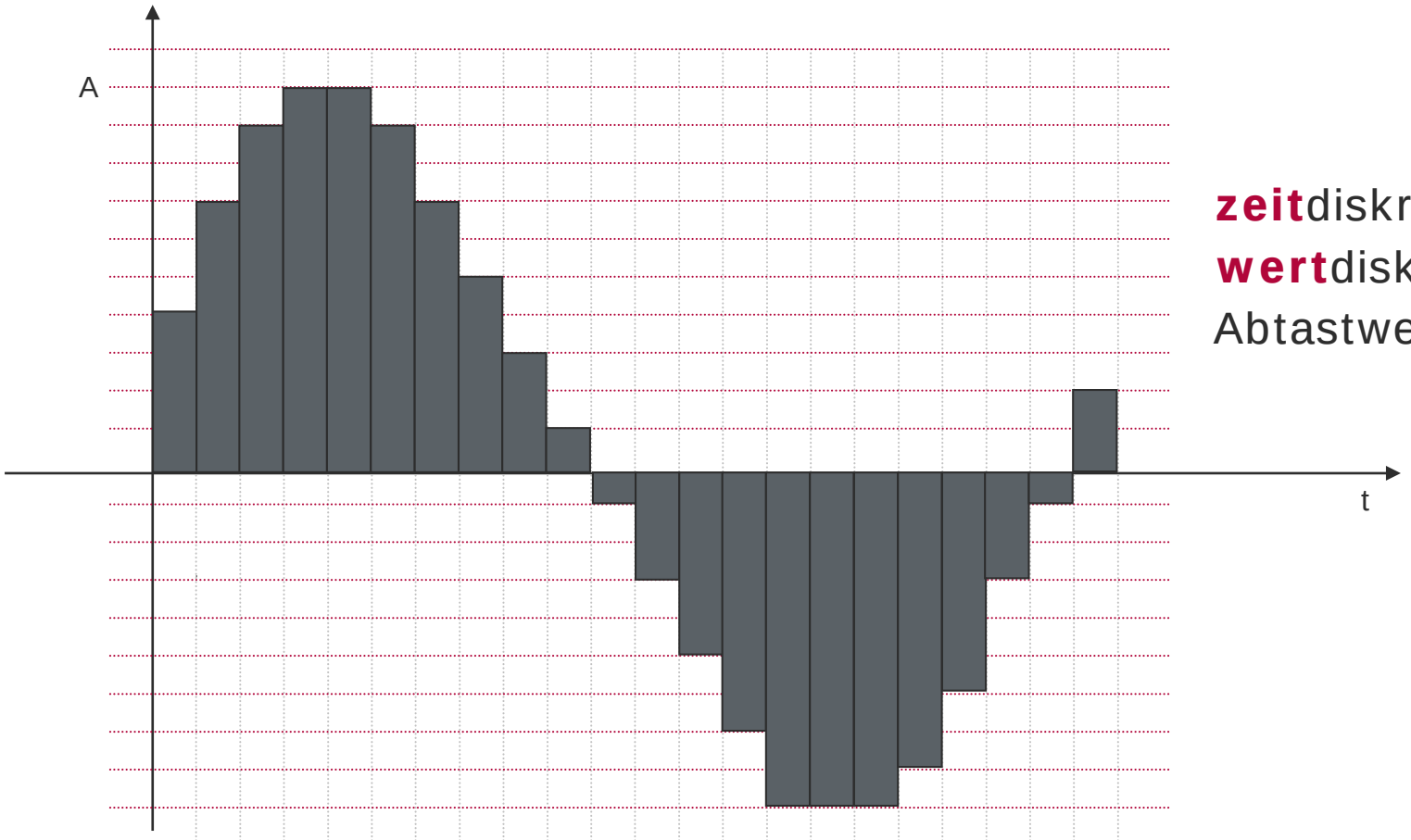


Audiokodierung und –komprimierung: Sampling und Quantisierung (4/6)



zeitdiskrete und
wertdiskrete
Abtastwerte

Audiokodierung und –komprimierung: Sampling und Quantisierung (5/6)



zeitdiskrete und
wertdiskrete
Abtastwerte

Audiokodierung und –komprimierung: Sampling und Quantisierung (6/6)

Problem:

- Wie viele Abtastpunkte?
- Wie viele Quantisierungsintervalle?

Ziel:

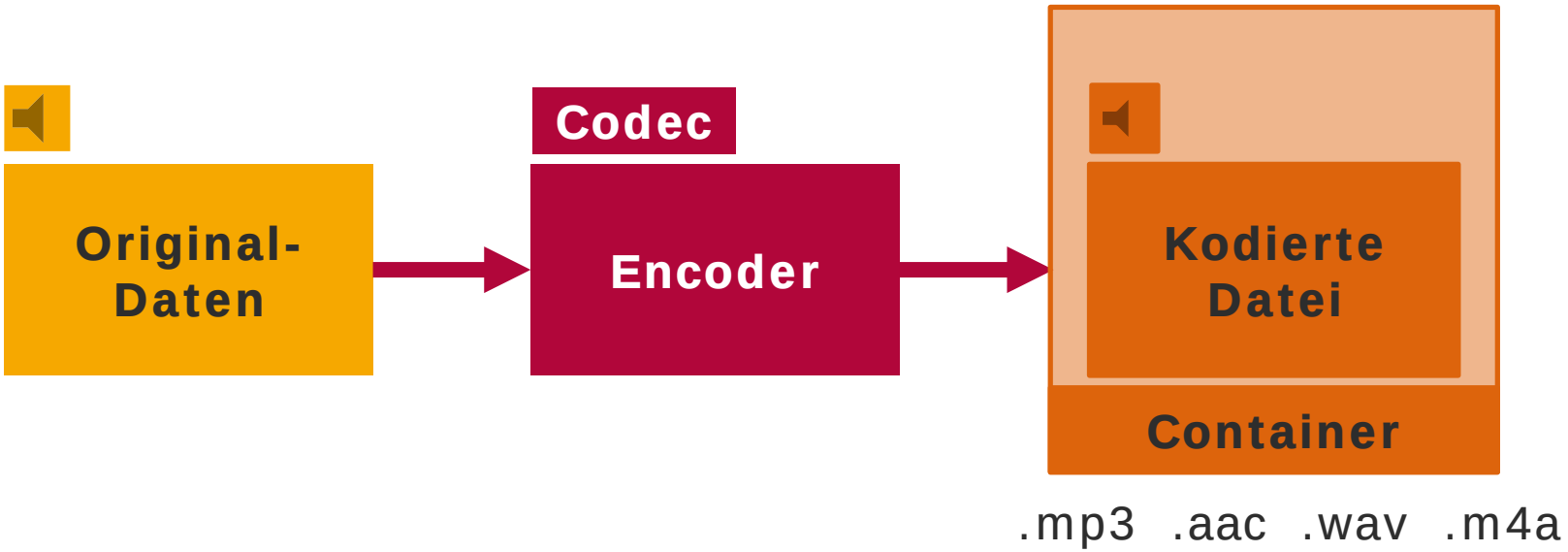
- Reproduktion des Ursprungssignals bei möglichst geringem Speicheraufwand

Verschiedene etablierte Verfahren, z.B. **PCM** – Pulsecode Modulation

Audiokodierung und –Komprimierung
Komprimierung

Digitalisierte analoge Audio-Information erfordert hohen Bedarf an Speicherplatz und Übertragungsbandbreite

- **Komprimierung** der Daten notwendig in ein standardisiertes Format, ggf. unter Inkaufnahme von vertretbarem **Informationsverlust**



Audiokodierung und –komprimierung: Formate

Unkomprimiertes Audio

- Sämtliche Audioinformation wird digital abgebildet (kodiert)
- Populäres Format: PCM in WAV-Container

Verlustbehaftetes komprimiertes Audio

- „Unnötige“ Information vermeiden, z.B.
  - nicht hörbare Geräusche eliminieren,
  - gleichartige Sequenzen zusammenfassen
  - ...
- Populäre Formate: MP3, AAC

Verlustfrei komprimiertes Audio

- (schwächere) Komprimierung von Audiosignalen ohne Verlust von Information
- Populäres Format: FLAC

openHPI: **Web-Technologien**

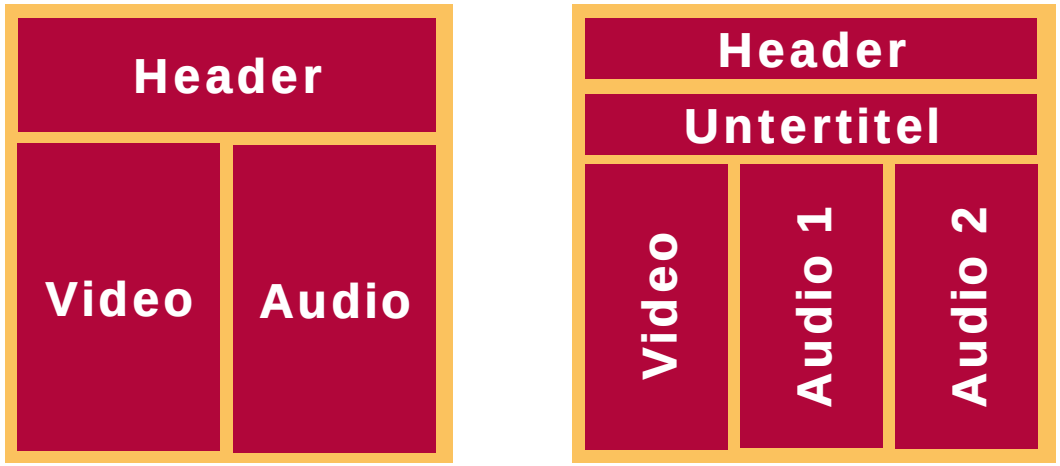
Videokodierung

Prof. Dr. Christoph Meinel
Hasso-Plattner-Institut
Universität Potsdam

Videokodierung

Typischerweise besteht Video aus bewegten Bildern und zugehöriger Audioinformation, die beide jeweils mit einem **Codec** komprimiert sind und in einem **Container** gespeichert werden

Es sind je Container auch mehrere Spuren und Untertitel speicherbar



- **Container**: MP4, AVI, WebM, OGV, MKV, ...
- **Videocodecs**: H.265, H.264, Ogg Theora, VP8, VP9, ...
- **Audiocodecs**: MP3, AAC, Ogg Vorbis, ...

Kompression von Videosignalen

Videomaterial verbraucht sehr viel Speicherplatz und macht einen Großteil der im WWW übertragenen Daten aus

Videosequenzen enthalten viel **Redundanz**:

- räumliche Redundanz
- zeitliche Redundanz

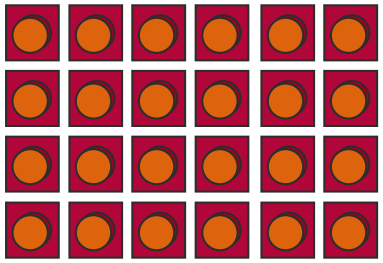
→ **Starkes Reduktionspotenzial** für Videokodierung

Allgemeines Vorgehen:

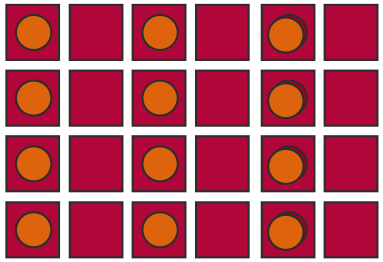
Subsampling (1/2)

Einsparung von Speicherplatz durch **Subsampling** (Unterabtastung)

→ Menschen können Farbinformationen nicht so genau wahrnehmen wie Helligkeitsinformation. Deshalb Speicherung von Helligkeit und Farbe nicht für jeden Pixel einzeln



4:4:4 – kein Subsampling



4:2:2 – horizontales Subsampling um Faktor 2

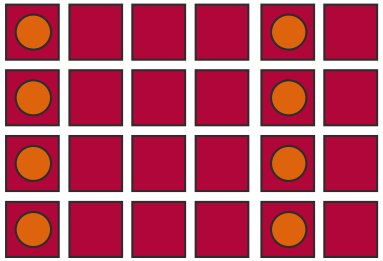
Luminanzpixel (Helligkeitsinformation)

Chrominanzpixel (Farbinformation)

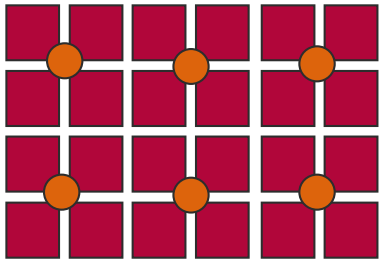
Subsampling (2/2)

Einsparung von Speicherplatz durch **Subsampling** (Unterabtastung)

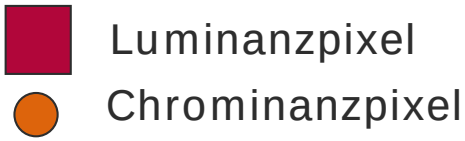
→ Menschen können Farbinformationen nicht so genau wahrnehmen wie Helligkeitsinformation. Deshalb Speicherung von Helligkeit und Farbe nicht für jeden Pixel einzeln



4:1:1 – horizontales Subsampling um Faktor 4



4:2:0 – horizontales und vertikales Subsampling um Faktor 2



Beispiel: HDTV

HDTV – High Definition Television

- Bildauflösung: bis 1920 x 1080 Pixel (Vollbild, kein Zeilensprung)
- Bildwiederholfrequenz: bis 60 Hz (Vollbild)
- Farbtiefe: 8 Bit
- Subsampling: 4:2:2
- Benötigte Bandbreite:

1920 x 1080 x 60 x 8 + 2x (960 x 1080 x 60 x 8) = **1,99 Gbps**

Luminanzpixel **Chrominanzpixel**

Beispiel für einen Videocodec: H.265

H.265 – HEVC (High Efficiency Video Coding)

- Nachfolger von H.264/MPEG-4 AVC
- Überlegungen zur Verbesserung von H.264 (wird u.a. bei Blu-ray Disc verwendet) und Schaffung eines Nachfolgers seit 2004
- 2013 wurde Standard HEVC/H.265 bestätigt
- Im Vergleich zu H.264
  - verbesserte Videoqualität bei doppelter Kompressionsrate
  - deutlich mehr Rechenlast beim Kodieren (Hardwareunterstützung muss noch besser werden)
- 8K UHD (Ultra HDTV) möglich, d.h. Auflösungen bis 8192 x 4320 (~17:9) – 35,4 Megapixel
- Neue Fernseher haben heute bereits H.265 integriert, weil H.265 u.a. beim neuen DVB-T2 (terrestrisches digitales Fernsehen) benutzt wird