FHD

2

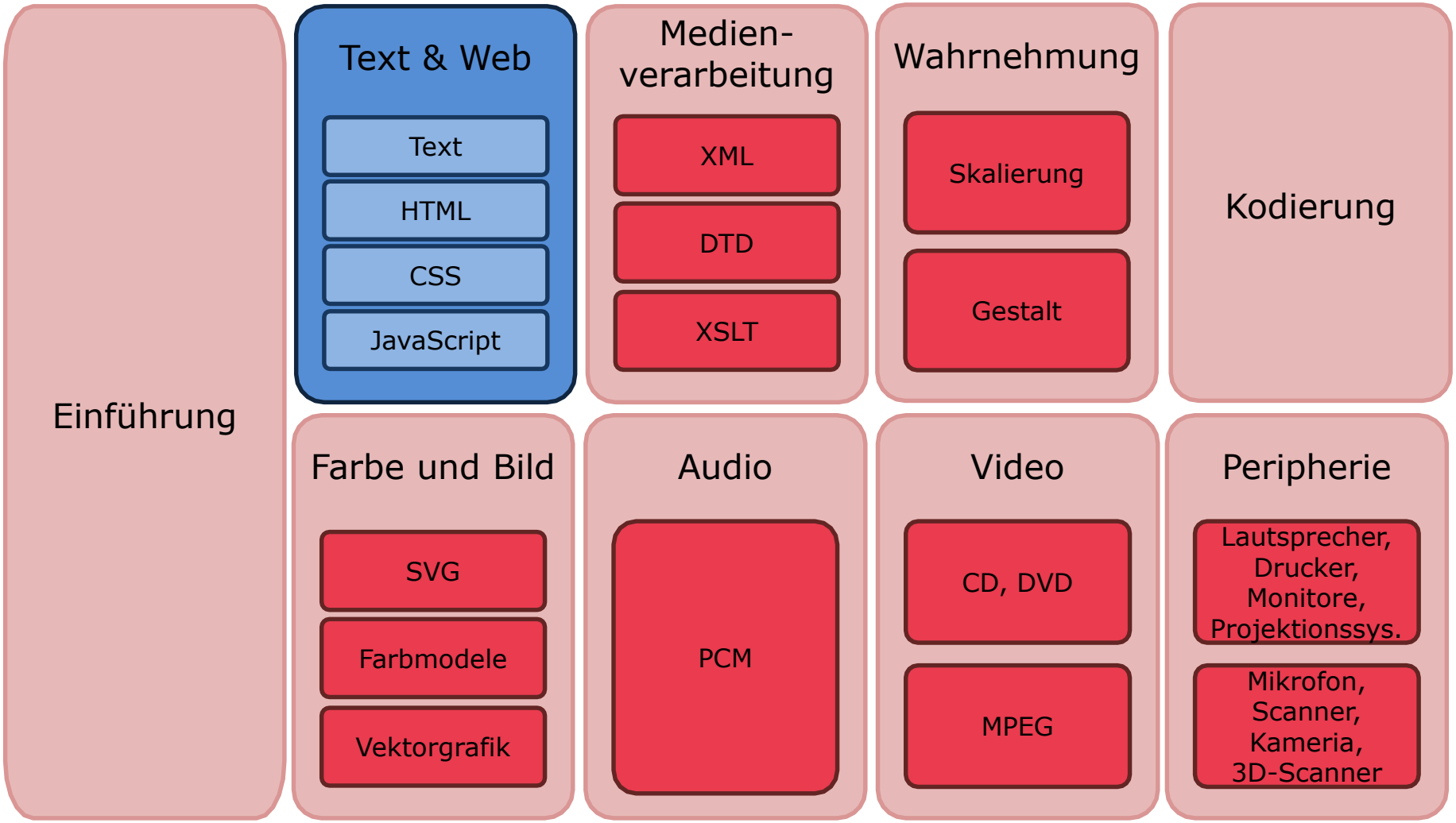
mit Material von G. Weber (TU Dresden):

Einführung in die Medieninformatik

Text und Web

Prof. Dr.-Ing. Tenshi Hara
fhd-emi@lern.es

VORLESUNGSÜBERSICHT



Text, Schrift und Formate

Zeichensatz

Ein Zeichensatz definiert eine Zuordnung zwischen Zeichen und Zahlen (feste Adressraumposition)

bspw. ASCII (American Standard Code for Information Interchange)

- 7-Bit-Zeichensatz (128 mögliche Zeichen)
- die Zeichen von 0 bis 31 sind Steuerzeichen (Zeilenvorschub, Tabulator, Löschen, …)
- die meisten Kode mit mehr als 7 Bits sind in den Kodezeichen 0 bis 127 identisch zum ASCII

Fremde Alphabete
zum Beispiel *Katakana*

ASCII (HEXADEZIMAL)

Kode	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
0_	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1_	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2_	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3_	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4_	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5_	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6_	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7_	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Unterscheidung zwischen Majuskeln und Minuskeln in einem Bit:

$$E \equiv 69_{10} = 45_{16} = 1\mathbf{0}00101_2$$

$$e \equiv 101_{10} = 65_{16} = 1\mathbf{1}00101_2$$

Zeichensatz-Kodeseiten

ISO 8859-x ($x \in [1; 16]$)

- *15* Varianten für 8-Bit Zeichensätze
- Kodezeichen 0 bis 127 identisch zum ASCII
- Latin-1 wird in weiteren Standards für Kodezeichen 0 bis 255 verwendet (Unicode)
- Deutsche Sonderzeichen in allen Latin-Varianten an selber Stelle
- Eurozeichen (€) in Latin-9
- Diese Norm wird nicht mehr weiterentwickelt

ISO 8859	
-1	Latin-1, Westeuropäisch
-2	Latin-2, Osteuropaisch
-3	Latin-3, Südeuropäisch
-4	Latin-4, Baltisch
-5	Kyrillisch
-6	Arabisch
-7	Griechisch
-8	Hebräisch
-9	Latin-5, Türkisch
-10	Latin-6, Norisch
-11	Thai
-12	Devanagari (nie umgesetzt)
-13	Latin-7, Baltisch
-14	Latin-8, Keltisch
-15	Latin-9, Westeuropäisch
-16	Latin-10, Südosteuropäisch

ISO/IEC 8859-15

Kode	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
0..1_	nicht definiert															
2..6_	identisch zum ASCII															
7_	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	*nicht def.*
8..9_	*nicht definiert*															
A_	NBSP	¡	¢	£	*€*	¥	*Š*	§	*š*	©	ª	«	¬	SHY	®	¯
B_	°	±	²	³	*Ž*	µ	¶	·	*ž*	¹	º	»	*Œ*	*œ*	*Ÿ*	¿
C_	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D_	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E_	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F_	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

rot und kursiv hervorgehoben: Unterschiede zu ISO 8859-1

UNICODE

Unicode-Standard

- internationaler Standard mit Ziel, alle Zeichen aus allen Nationen zu kodieren
- Kodierung in der Praxis: hexadezimal mit vorangestelltem „U+“ (Universal Character Set)
- UCS-2 umfasst 16 Bit je Kodezeichen
- UCS-4 umfasst $2^{20}+2^{16}$ = 1.114.112 Zeichen

Unicode-Formate

sowohl Little Endian (LE) als auch Big Endian (BE) möglich

- *UTF-32* … vier Bytes pro Zeichen
- *UTF-16* … fast immer zwei Bytes pro Zeichen; Kennzeichnung der weiteren Zeichen durch vier Bytes mit speziellem Code in den ersten zwei Bytes (Surrogates)
- UTF-8 … ein Byte pro Zeichen für ASCII-Zeichen; ist höherwertiges Bit gesetzt, werden 2 oder 4 Bytes verwendet

Aufbauschema für UTF-8

Das erste Byte eines UTF-8-kodierten Zeichens ist Start-Byte, weitere Bytes sind Folge-Bytes.

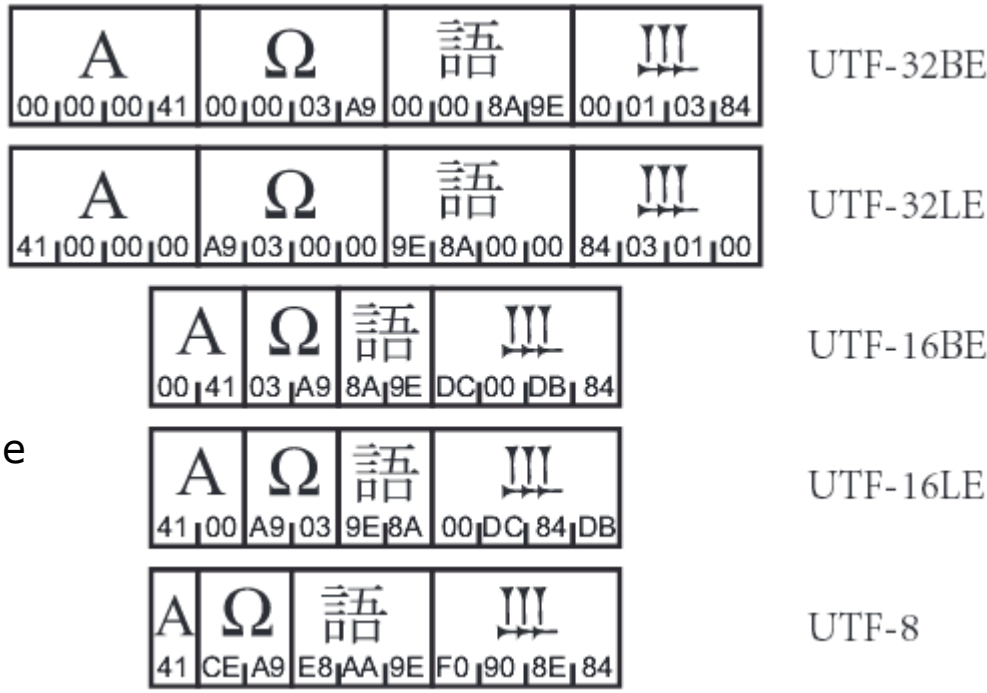
Start-Bytes beginnen* immer mit den Bits `0` oder `11`, Folge-Bytes mit den Bits `10`.

Wenn Start-Byte mit dem Bit `0` beginnt*, handelt es sich um ein 1bytiges Zeichen.
→ vollständig zum ASCII kompatibel.

Wenn Start-Byte mit dem Bit `1` beginnt*, handelt es sich um ein Mehrbytezeichen. Alle weiteren Bytes (Folge-Bytes) müssen mit den Bits `10` beginnen*, sonst fehlerhaft.

Die Anzahl führender 1en im Start-Byte ist gleich der gesamten Byte-Anzahl des Mehrbyte-Zeichens.
→ Beispiel: `1110xxxx10xxxxxx10xxxxxx`

Vergleich der UTF-Kodierungen:



*: LE und BE beachten!

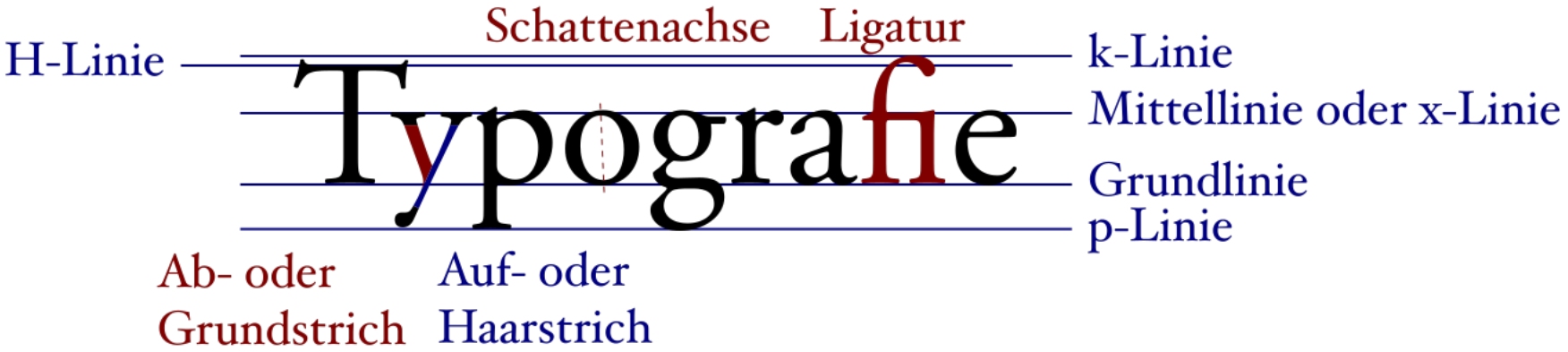
Braille

Leerzeichen	A oder 1	,	B oder 2	.	K	;	L	"	C oder 3	I oder 9	F oder 6	ÄU	M	S	P
~	E oder 5	:	H oder 8	*	O	! oder +	R	>	D oder 4	J oder 0	G oder 7	Ä	N	T	Q
'	AU	?	EU	-	U	»	V	$	EI	Ö	`	IE	X	ß	&
	SCH	/	Ü	«	Z	(,) oder =	[	_	CH	W	^	#	Y	ST oder]	%

Im deutschen Standardalphabet bleibt die unterste Reihe der Punktmatrix unbesetzt.
(d.h. 192 nicht verwendete Symbole)

- Blindenschrift
- Punktschrift mit bis zu 256 Symbolen, teilweise doppelt belegt
- für (fast) alle Sprachen verfügbar
- Kodierung in 6 (Kurzform, 64 Hauptsymbole) oder 8 möglichen Punkten (256 Symbole)
- hoher Platzbedarf (Duden: 18 Bänden; Neues Testament: 42 Bänden)
  - Punkte müssen haptisch wahrzunehmen sein
  - Platzersparnis durch Voll-, Kurz-, Steno-, Mathematik-, Chemie- und Musikschriften

Grundlegende Begriffe der Schriftartbeschreibung



Maßeinheiten bei Schriften

- Historische Maße
  - Truchet: 1pt = 0,188mm
  - l'imprimerie nationale (IN): 1pt = 0,4mm
- Didot
  - 1pt = 0,376mm = $\frac{1}{72}$ RF Pouce
  - 1cc = 4,531mm = 12pt
- ATA
  - 1pt = 0,3514598mm = 0,013837"
  - 1pc = 4,2175176mm = 12pt
- TeX
  - 1pt = 0,3514598035mm = $\frac{1}{72{,}27}$"
  - 1pc = 4,217517642mm = 12pt
- DTP
  - 1pt = 0,3527777778mm = $\frac{1}{72}$"
  - 1pc = 4,233333333mm = 12pt

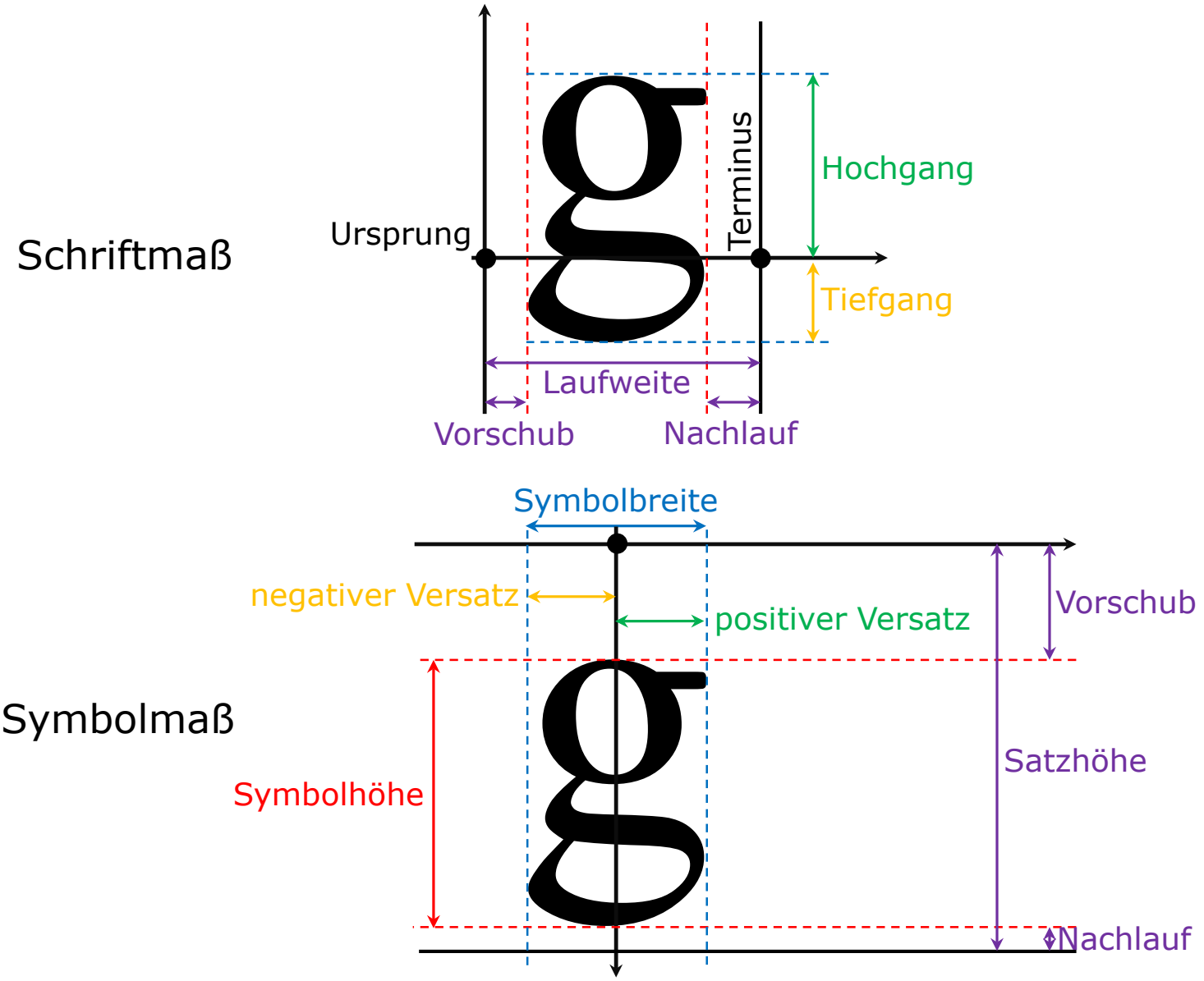
» statt „*DTP Punkt*" (Desktop Publishing) oft „*Postscript Punkt*"

» verwendete Abkürzungen:
pt (point), dd (Didot point), pc (pica), cc (cicero), " (Zoll, Inch)

Grundlegende Eigenschaften von Schriften

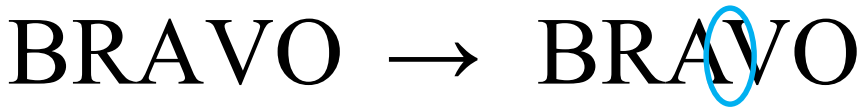
- Zeichencodierung
- Serifen helfen der Augenführung, sollten jedoch nur bei Fließtexten eingesetzt werden.
- Schriftschnitt/Schriftstil
  - Schriftstärke (mager, normal, fett, …)
  - Schriftbreite (schmal, normal, breit, …)
  - Schriftlage (normal, kursiv, …)
- Symboleigenschaften
  - Symbolabstand, Kerning, Unterschneidung (äquidistant, flexibel, …)
  - Symbolbreite (einheitlich, proportional, gruppiert)
- Computerschriften unterteilt in
  - Vektorschriften: Definition der Umrisse durch Kurven
  - Rasterschriften: ein Bild je Symbol

Schrift- und Symbolmetrik



KERNING/UNTERSCHNEIDUNG

- Anpassung von Zeichenabständen bei parallel Diagonalen

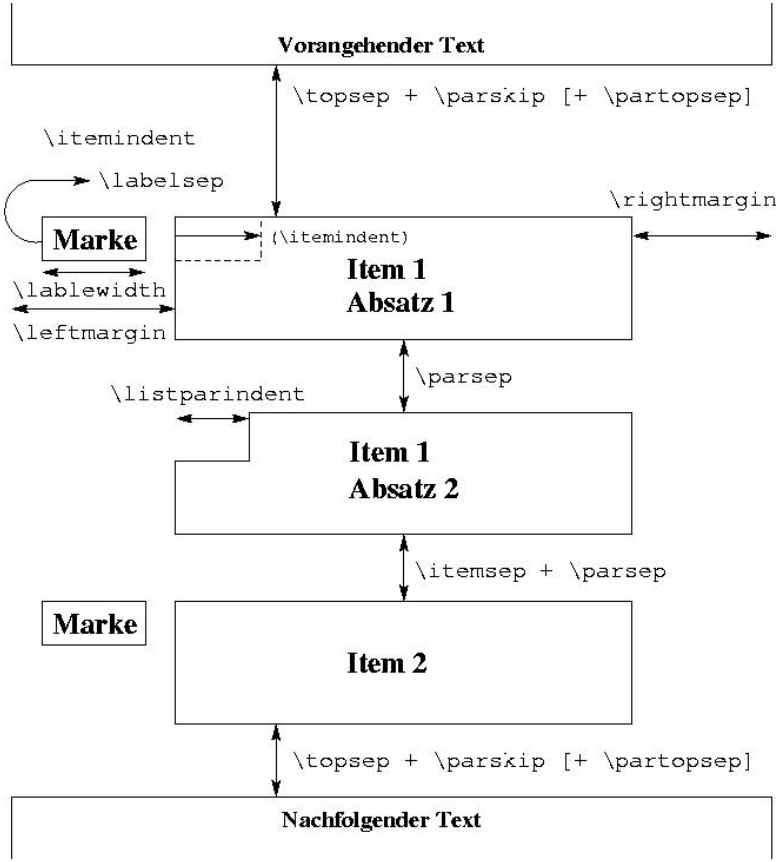


- korrigierter Abstand wird pro Buchstabenpaar angegeben
- Vorsicht bei sehr kurzer Zeichenbreite:

ABSÄTZE

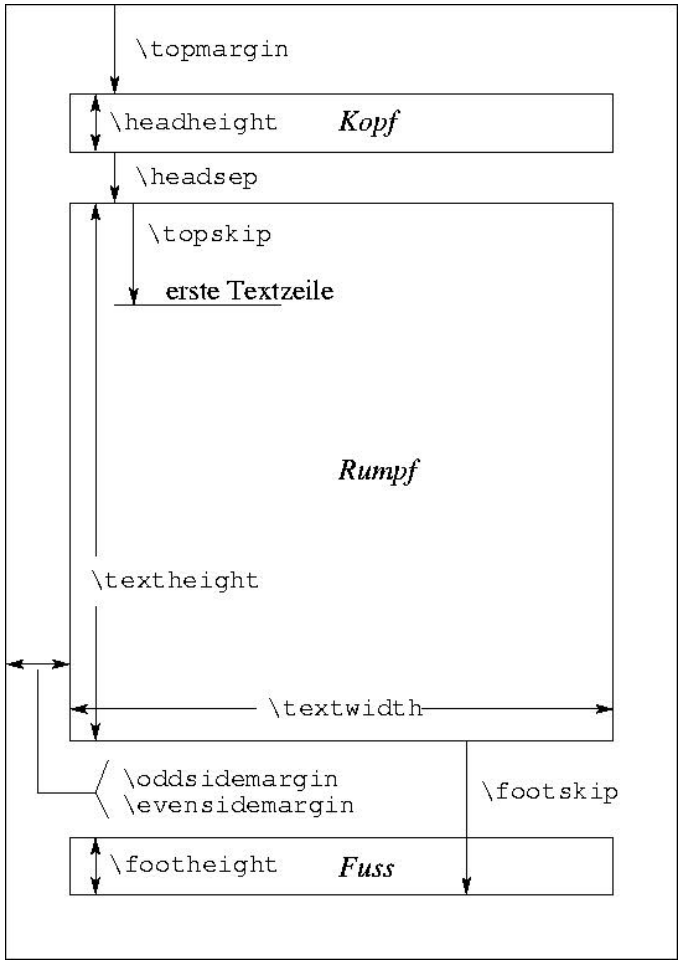
- Setzen von Absätze
- Absatzformat
  - Abstände
  - Einrücken
- Blocksatz durch Variation von Wortabständen
- Flattersatz und Mittelachsensatz durch variable Wortabstände
- automatische Silbentrennung

Beispiel in LaTeX:



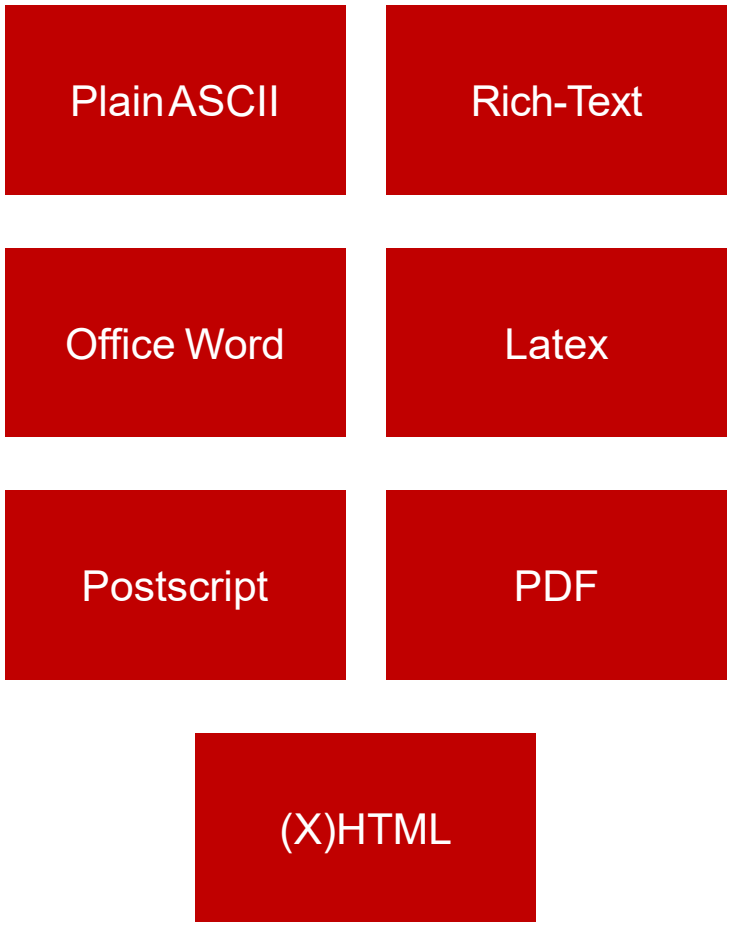
SEITENLAYOUT

Beispiel in LaTeX:



- Fußnoten
- Verzeichnisse
  - Inhalt
  - Stichworte
  - Literatur
  - Abbildungen
  - …
- Verweise
- Formeln

Formate und Software



Eigenschaften

Editierbarkeit

WYSIWYG

Lesbarkeit des Formats

Integration von Medien

Funktionsumfang

Programmierbarkeit

Rich-Text Format (RTF)

- 1987 von Microsoft offengelegter Standard
- MS Word als WYSIWYG-Editor
- Format in Grenzen lesbar
- eigene Repräsentation für Bildern und Vektorgraphiken
- nicht programmierbar

Beispiel

RTF-Inhalt:

```
{\rtf
   Guten Tag!
   \par
   {\i Dies} ist ein
      kleiner {\b Text}.
   \par
   Das Ende.
}
```

Darstellung:

Guten Tag!

Dies ist ein kleiner **Text**.

Das Ende.

LaTeX

- 1984 von Lamport entwickelte Interface-Sprache für TeX (Textsatz-System)
- nicht für WYSIWYG gedacht; erzeugt DVI, PS, PDF, HTML, …
- Medien über Verweise
- voll programmierbar
- frei verfügbar

Diskussion

✓ sehr schöner Textsatz

✓ sehr schöne Formeln

✓ sehr gute Literatur- und Verzeichnisfunktionen

✓ sehr große Dokumente handhabbar; auch in Teilen

✓ volle Transparenz

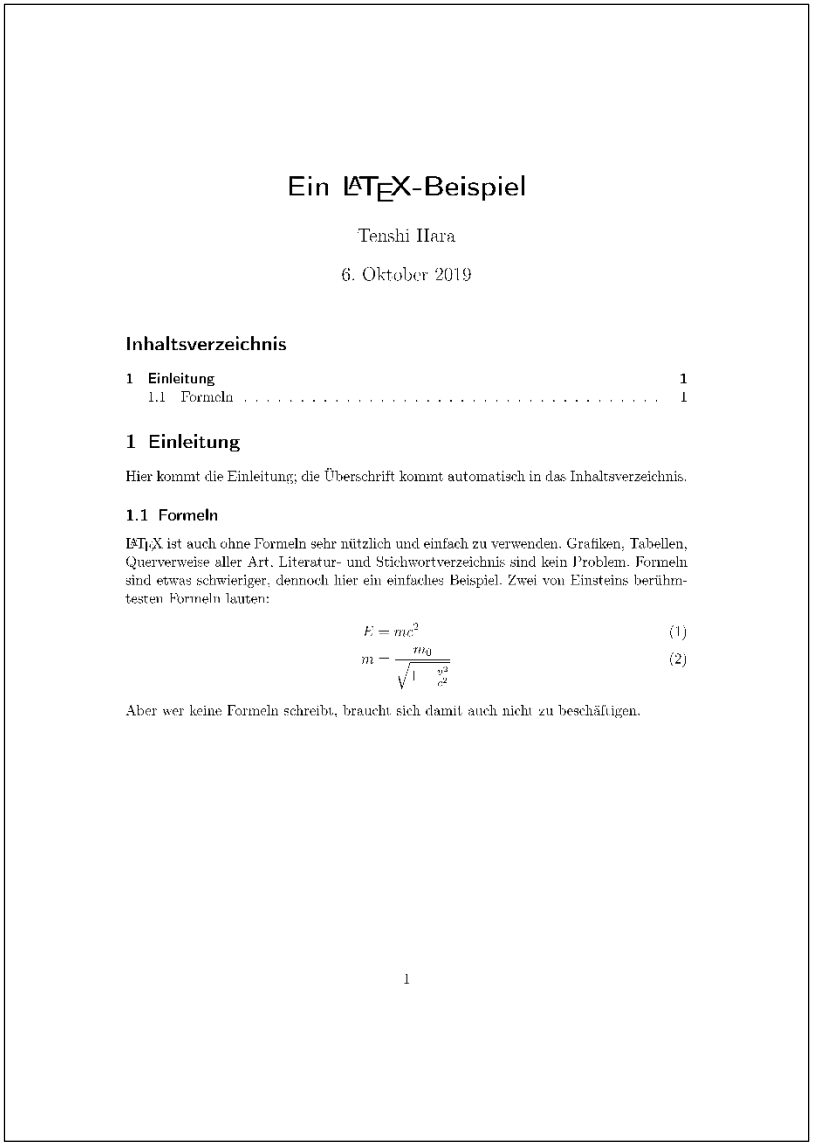
– längere Einarbeitungszeit; sehr steile Lernkurve

– bereits bei kleineren Dokumenten längere Entwicklungszeiten

– Makros erhöhen Komplexität

Text – Formate und Software – LaTeX (2)

```
%% Erläuterungen zu den Befehlen erfolgen unter
%% diesem Beispiel.
\documentclass{scrartcl}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[ngerman]{babel}
\usepackage{amsmath}
\title{Ein \LaTeX{}-Beispiel}
\author{Tenshi Hara}
\date{6. Oktober 2019}
\begin{document}
\maketitle
\tableofcontents
\section{Einleitung}
Hier kommt die Einleitung; die Überschrift kommt
automatisch in das Inhaltsverzeichnis.
\subsection{Formeln}
\LaTeX{} ist auch ohne Formeln sehr nützlich und
einfach zu verwenden. Grafiken, Tabellen,
Querverweise aller Art, Literatur- und
Stichwortverzeichnis sind kein Problem. Formeln
sind etwas schwieriger, dennoch hier ein einfaches
Beispiel.    Zwei von Einsteins berühmtesten
Formeln lauten:
\begin{align}
    E &= mc^2\\
    m &= \frac{m_0}{\sqrt{1-\frac{v^2}{c^2}}}
\end{align}
Aber wer keine Formeln schreibt, braucht sich damit
auch nicht zu beschäftigen.
\end{document}
```



Ein LaTeX-Beispiel

Tenshi Hara

6. Oktober 2019

Inhaltsverzeichnis

1 Einleitung

Hier kommt die Einleitung; die Überschrift kommt automatisch in das Inhaltsverzeichnis.

1.1 Formeln

LaTeX ist auch ohne Formeln sehr nützlich und einfach zu verwenden. Grafiken, Tabellen, Querverweise aller Art. Literatur- und Stichwortverzeichnis sind kein Problem. Formeln sind etwas schwieriger, dennoch hier ein einfaches Beispiel. Zwei von Einsteins berühmtesten Formeln lauten:

$$E = mc^2 \quad (1)$$

$$m = \frac{m_0}{\sqrt{1 - \frac{v^2}{c^2}}} \quad (2)$$

Aber wer keine Formeln schreibt, braucht sich damit auch nicht zu beschäftigen.

1

Portable Document Format (PDF)

- von Adobe entwickelt und 1993 publik gemacht
- PDF baut auf Adobe Postscript auf
- umfangreiches Rechtemanagement und Verschlüsselungsoptionen
- durch kostenfreien Reader zum de-facto-Standard geworden
- ist Layout-treu → Dokumente werden immer gleich wiedergegeben
  - dennoch Einbettung von Schriften problematisch
  - kann nur bedingt mit speziellen Werkzeugen editiert werden
- alle gängigen Medientypen (auch 3D-Graphik) können integriert werden
- Erzeugung aus Postscript (Distiller), LaTeX (pdflatex), HTML (htmldoc), … (unter Windows über (obfuskierten) Druckertreiber)

Hypertext Markup Language (HTML)

Hypertext Transfer Protocol (HTTP)

- Spezifikation der Kommunikation zwischen Web-Server und Web-Client
  - Versand der HTML Dateien vom Web-Server an den Web-Browser
  - Versand der Formulareingaben vom Web-Browser an den Web-Server
- Spezifikation der Nutzung weiterer Technologien
  - TCP Connection Keep Alice
  - TLS-Verschlüsselung
  - WebSockets
  - …

Konzepte des World Wide Web

Universal Resource Identifier (URI)

- Bezeichnung und Adresierung beliebiger Ressourcen im Web
- URIs unterteilen sich in URL (Uniform Resource Locator) und URN (Uniform Resource Name)

Hypertext Markup Language (HTML)

- Markup-Sprache zur Beschreibung von Web-Dokumenten
- definiert den Inhalt und die Gliederung von (Text-)Ressourcen

Cascading Style Sheets (CSS)

- Auszeichnungssprache zur Beschreibung von Gestaltungsmustern
- definiert die Gestaltung von (HTML-)Ressourcen

HYPERTEXT

Hypertext ist ein nichtlineares Medium zur Präsentation von Textinhalten. Die Nichtlinearität wird durch Hyperlinks erzielt, die verschiedene Einheiten verknüpfen.

Bush entwickelte 1945 eine technisch-wissenschaftliche Utopie (Memex), die darauf basiert, dass Denken durch Assoziationen gefördert wird.

Das Konzept sieht vor, dass riesige Informationsmengen gespeichert werden können, aber auch dass Benutzer Spuren anlegen, verwandte Texte verknüpfen und die wiederum für den zukünftigen Einsatz verwendet werden können.

Der Begriff *Hypertext* wurde 1965 vom Autor und Philosophen Ted Nelson entwickelt. (siehe auch http://ted.hyperland.com/)

Hypertext Markup Language (HTML)

- Grundlage des WWW
- Einteilung in
  - darzustellende Inhalte
  - zusätzliche Metainformationen
- verwandt mit XML
  - HTML 4.01 (XHTML) vollständig kompatibel mit XML
  - HTML 5 ersetzt XML-ähnliches HTML und XHTML durch XML-valide Beschreibung
- visuelle Darstellung soll nicht in HTML erfolgen
  (sondern durch den Web-Browser und Gestaltungsvorlagen wie CSS)
- HTML ist eine Auszeichnungssprache, keine Programmiersprache
- ähnliches, um Textsatz für Printmedium ergänztes Konzept: LaTeX

Syntax (1/2)

- textbasierte Auszeichnungssprache zur semantischen Strukturierung elektronischer Dokumente
  - Texte mit Hyperlinks
  - Medieninhalte
- Auszeichnungen (Markup) von Textteilen zur Strukturierung
- Markups werden durch Tags markiert (Einzel-Tag oder Tag-Paar)
  - Tags beginnen mit < und enden mit >
  - End-Tags beginnen mit </
  - Einzel-Tags beginnen mit < und enden mit />
- Tags enthalten den Elementnamen und eine optionale Attributliste
- zusammengehörenden Start- und End-Tags bilden mit dem dazwischenliegenden Inhalt ein Element
- Elemente sind verschachtelbar (siehe DTD)

Syntax (2/2)

- bestimmte Elemente müssen nicht explizit notiert werden (HTML4)
- einigen Elementen darf End-Tag fehlen (`<p>`, `<li>`, `<dd>`, `<br>`, …) (HTML4)
- Groß- und Kleinschreibung der Tags irrelevant (HTML4)
- inhaltsleere Elemente möglich, müssen einelementig schließen (`<br />`, `<hr />`, ...) (XHTML, HTML5)
- darstellungsorientierte Attribute möglich (`align="…"`, …) (HTML4, XHTML)
- darstellungsorientierte Attribute verboten (stattdessen `style="…"` (unerwünscht) oder CSS (bevorzugt)) (HTML5)
- Elemente werden strukturieret ausgezeichnet
  - Textbereichen wird Bedeutung zugeordnet (`<h1>`Überschrift`</h1>`, `<p>`Absatz`</p>`, `<em>`betonter Text`</em>`, ...)
  - HTML gibt Anweisungen für das Parsen
    - Quelltextes einlesen und Informationen verarbeiten: Parsen
    - Aufbereitung für das Ausgabemedium: Rendern

Zeichenvorrat

- ursprünglich 7-Bit-ASCII
- ergänzt um zahlreiche, als HTML-Entität kodierte Sonderzeichen (bspw. ä kodiert als `&auml;`, `&#228;`, `&#x00E4;`)
- Unterstützung universeller Zeichensätze erst mit UTF
- abwärtskompatibel auf plattformunabhängige Portierbarkeit angelegt
- Wahl des zugrundeliegenden Zeichenvorrats sollte in Meta-Angaben für Browser hinterlegt sein
- Achtung! URLs werden nach MIME-Verfahren in ASCII-Zeichen kodiert (`%20` für Leerzeichen, `%E4` für ä, ...)

Wichtige Tags

• <html>…</html>	Wurzelelement
• <head>…</head>	Dokumentkopf mit Meta-Infos
• <title>…</title>	Dokumenttitel
• <meta name="…" content="…" />	Meta-Information
• <body>…</body>	Dokumentinhalt
• … , …, …	Überschriften
• …	Absatz
• 	erzwungener Zeilenumbruch
• <hr />	Trennlinie
• 	Bild
• …	Inhalts-Container
• <script>…</script>	Client-seitig zu kompilierende und auszuführende Programme

Anker

- wichtigstes Element in HTML
  → macht aus gewöhnlichem Text Hypertext

- definiert adressierbare Anker (Anchor) im Quelltext
  → entsprechen Positionen im angezeigten Dokument

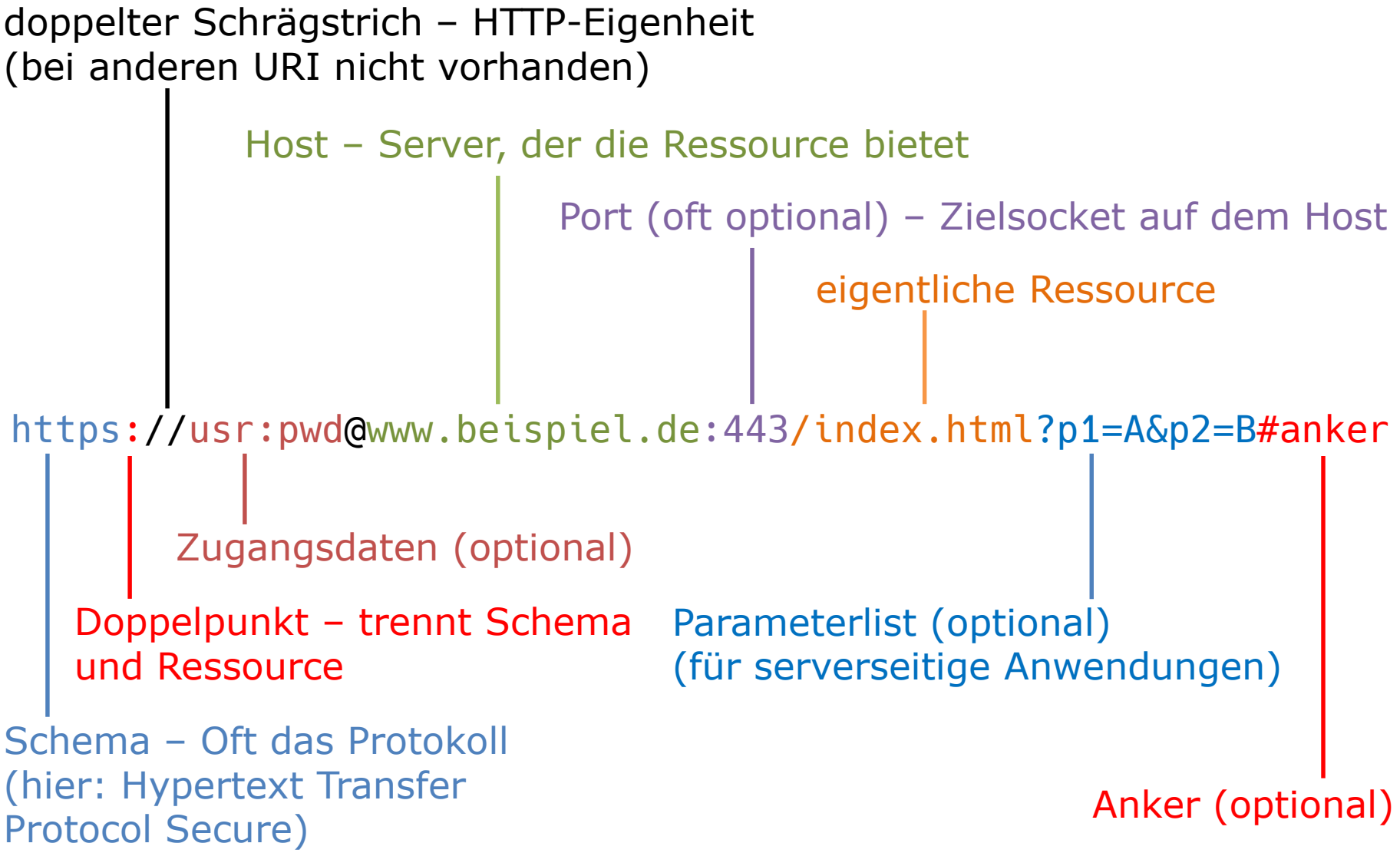
```
<a name="ankermarke">Anzeigetext</a>
```

- setzt Sprungziele (Link) zu anderen Dokumenten oder Ankern
  → oft als Uniform Resource Identifier (URI) oder Locator (URL)

```
<a href="schema:ressource" target="ziel">Anzeigetext</a>
```

  - unterschiedliche Protokolle und Ressourcentypen als Ziel in der Hypertextreferenz (`href`) möglich
  - angeforderte Ressource kann (optional) mittels `target` an ein Ziel gegeben werden (`_self`, `_blank`, `_parent`, `_top`, `Framename`)

URI und URL


doppelter Schrägstrich – HTTP-Eigenheit
(bei anderen URI nicht vorhanden)
Host – Server, der die Ressource bietet
Port (oft optional) – Zielsocket auf dem Host
eigentliche Ressource
https://usr:pwd@www.beispiel.de:443/index.html?p1=A&p2=B#anker
Zugangsdaten (optional)
Doppelpunkt – trennt Schema
und Ressource
Parameterlist (optional)
(für serverseitige Anwendungen)
Schema – Oft das Protokoll
(hier: Hypertext Transfer
Protocol Secure)
Anker (optional)

ABSOLUTE UND RELATIVE URL

Beispiele für absolute URL

`https://de.wikipedia.org/wiki/URL#Beispiele`

`ftps://pi:raspberry@172.20.209.100:22/~/Downloads/`

`mailto:fhd-emi2019@lern.es`

`file:///F:/Lehre/GLWEP/1%20-%20W3C-Standards.pptx`

Beispiele für relative URL

`//de.wikipedia.org/wiki/Datei:Vivaldi.png` – gleiches Protokoll

`/wiki/Vivaldi_(Browser)` – gleiches Protokoll, gleicher Host (entspricht absolutem Pfad)

`Vivaldi_(Browser)` – gleiche Pfadebene

`#Versionen` – Anker in der gleichen Ressource

`#` – die Ressource selbst (oft bei jQuery, da href im a-Tag nicht leer sein soll)

Hallo-Welt-Web-Seite (HTML 5)

```
<!DOCTYPE html>
<html>
   <head>
      <title>Hallo Welt!</title>
      <meta name="author" content="Tenshi Hara" />
      <meta http-equiv="content-language" content="de" />
   </head>
   <body>
      <h1>Hallo Welt!</h1>
      <p>Das ist meine erste Webseite.</p>
      <hr />
   </body>
</html>
```

kann in den gängigen Browsern weggelassen werden
(für absolut konforme Dokumente aber notwendig!)

Hallo-Welt-Web-Seite (XHTML 1.1)

bei XHTML notwendig!

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
   <head>
      <title>Hallo Welt!</title>
      <meta name="author" content="Tenshi Hara" />
      <meta http-equiv="content-language" content="de" />
   </head>
   <body>
      <h1>Hallo Welt!</h1>
      <p>Das ist meine erste Webseite.</p>
      <hr />
   </body>
</html>
```

Formulare

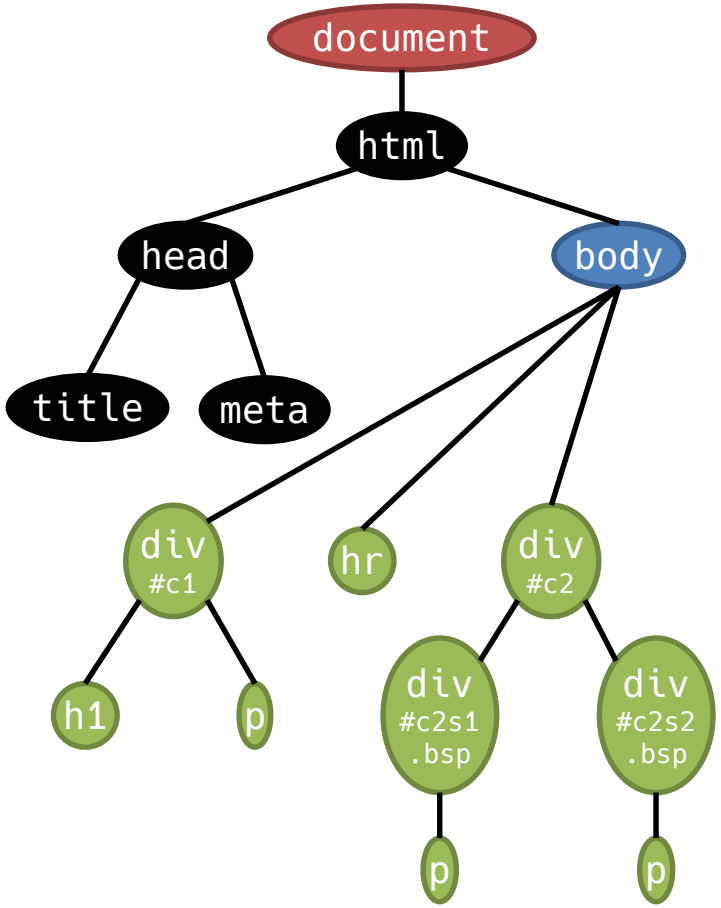
- Schlüsselelement zum Senden von Daten an den Server
- zwei grundlegende Möglichkeiten zum Senden
  - GET Daten werden in der URI als Parameter übertragen
    (z.B. `https://example.com/form.html?p1=Test&p2=null`)
  - POST Daten werden im Header übertragen
    (in Kommunikationsprotokollen erscheinen sie nicht in der URI)
- grundlegender Tag: `<form>`
  - innerhalb jedes Formulars gibt es Eingabe-Elemente (`<input>`) mit verschiedenen Typen (Text, Zahl, Datum, Datei, …)
  - mindestens einen Mechanismus zum Absenden
    (i.d.R. `<input type="submit">` oder JavaScript `form.submit()`)
  - ggf. einen Mechanismus zum Zurücksetzen
    (z.B. `<input type="reset">`)

Document Object Model (DOM)

Document Object Model (DOM)

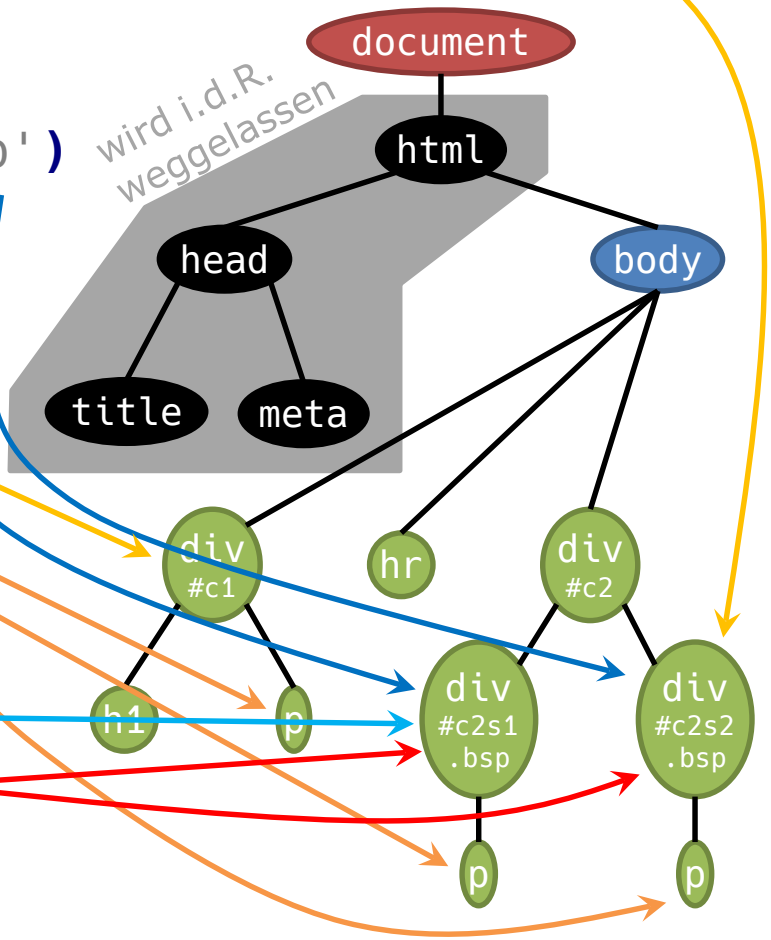
- alle HTML-Dokumente lassen sich anhand ihrer Struktur in einen Baum überführen, nämlich das Objektmodell des Dokuments (DOM)

```
<html>
  <head>
    <title>Hallo Welt! - Version 2</title>
    <meta name="author" content="Tenshi Hara" />
  </head>
  <body>
    <div id="c1">
      <h1>Hallo Welt!</h1>
      <p>Sei gegr&uuml;&szlig;t, liebe Welt.</p>
    </div>
    <hr />
    <div id="c2">
      <div id="c2s1" class="bsp">
        <p>DOM-Beispiel</p>
      </div>
      <div id="c2s2" class="bsp">
        <p>Im Rahmen von 3MI-GLWEB-20</p>
      </div>
    </div>
  </body>
</html>
```



DOM-Zugriffe

- vollständiger Pfad:
  `document.body.div[2].div[2]`
- surjektive Klasse:
  `document.getElementsByClassName('bsp')`
- bijektiver Identifier:
  `document.getElementById('c1')`
- Element-Typ:
  `document.getElementsByTagName('p')`
- Query-Selektoren
  `document.querySelector('#c2s1')`
  `document.querySelectorAll('.bsp')`

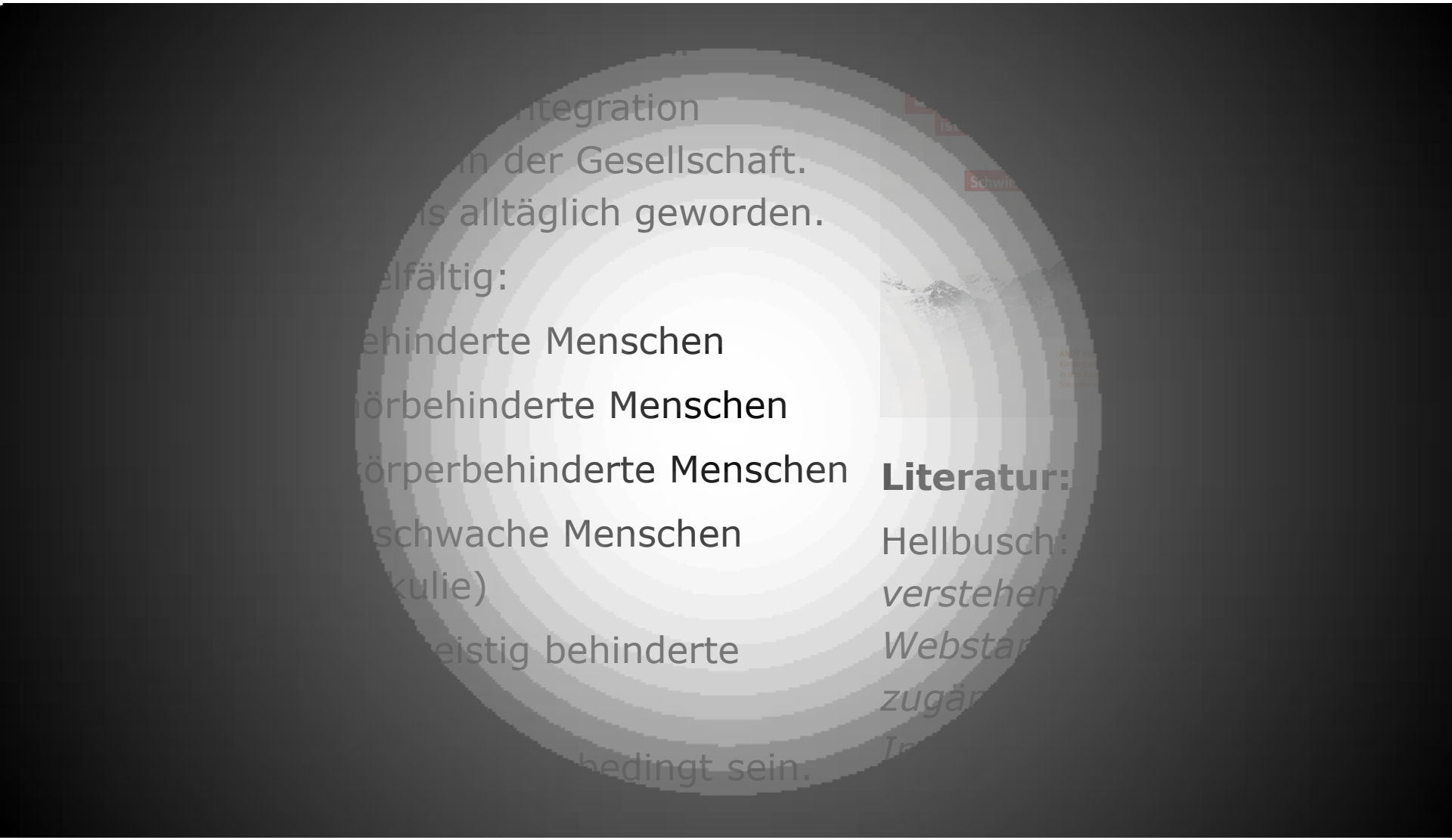


Exkurs: Barrierefreiheit im Web

Literatur:

Hellbusch:
verstehen
Webstan
zugän
In

Eingeschränktes Sichtfeld



EINGESCHRÄNKTE FARB- UND KONTRAST WAHRNEHMUNG

Medien können Barrieren erzeugen.

Barrieren verhindern die Integration behinderter Menschen in der Gesellschaft.
→ Das Web ist für uns alltäglich geworden.

Behinderung ist vielfältig:

- blinde und sehbehinderte Menschen
- gehörlose und hörbehinderte Menschen
- motorisch und körperbehinderte Menschen
- lese- und rechenschwache Menschen (Dyslexie, Dyskalkulie)
- lern-, sprach- und geistig behinderte Menschen

Behinderungen können altersbedingt sein.

Literatur:

Hellbusch: *Barrierefreiheit verstehen und umsetzen: Webstandards für ein zugängliches und nutzbares Internet*; dpunkt Verlag

GRAUER STAR

Medien können Barrieren erzeugen.

Barrieren verhindern die Integration behinderter Menschen in der Gesellschaft.

→ Das Web ist für uns alltäglich geworden.

Behinderung ist vielfältig:

- blinde und sehbehinderte Menschen
- gehörlose und hörbehinderte Menschen
- motorisch und körperbehinderte Menschen
- lese- und rechenschwache Menschen (Dyslexie, Dyskalkulie)
- lern-, sprach- und geistig behinderte Menschen

Behinderungen können altersbedingt sein.

Literatur:

Hellbusch: *Barrierefreiheit verstehen und umsetzen: Webstandards für ein zugängliches und nutzbares Internet*; dpunkt Verlag

Barrieren verstehen

Medien können Barrieren erzeugen.

Barrieren verhindern die Integration behinderter Menschen in der Gesellschaft.
→ Das Web ist für uns alltäglich geworden.

Behinderung ist vielfältig:

- blinde und sehbehinderte Menschen
- gehörlose und hörbehinderte Menschen
- motorisch und körperbehinderte Menschen
- lese- und rechenschwache Menschen (Dyslexie, Dyskalkulie)
- lern-, sprach- und geistig behinderte Menschen

Behinderungen können altersbedingt sein.

Literatur:

Hellbusch: *Barrierefreiheit verstehen und umsetzen: Webstandards für ein zugängliches und nutzbares Internet*; dpunkt Verlag

Cascading Style Sheets (CSS)

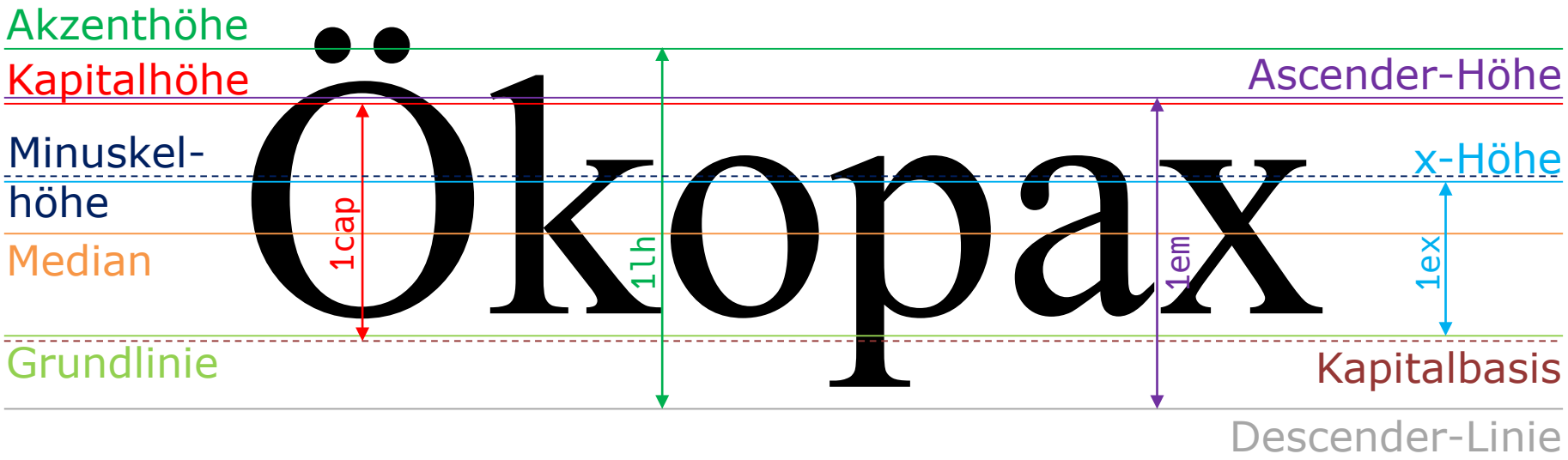
Sinn und Zweck

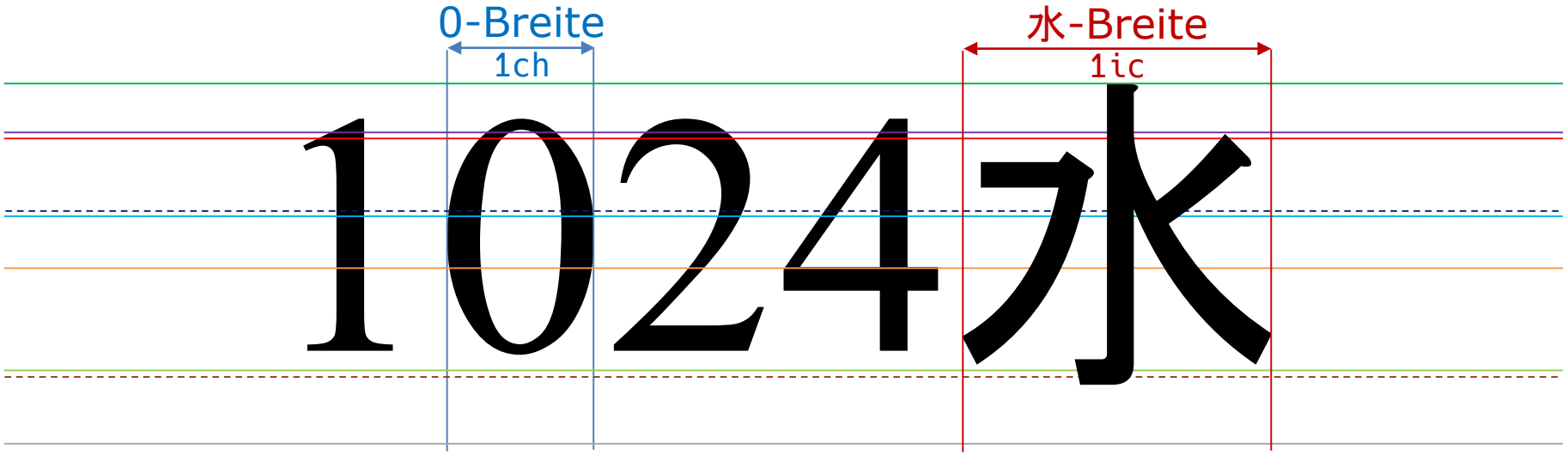
- Auszeichnungssprache für (X)HTML u.a. Markup-Sprachen zur Gestaltung des definierten Inhalts → klare Trennung von Präsentation und Inhalt
- fördert die Barrierefreiheit
  (bspw. durch wählbare Vordergrundfarbe und Hintergrundfarbe)
  - Bereitstellung barrierefreier Design
  - Überschreiben des Designs durch Client-seitige Voreinstellungen
- fördert breite Geräteunterstützung
  → Responsivität durch CSS Media Queries

Einbinden von CSS

- kann im HTML-Dokument selbst definiert werden
  - im Header oder Body mittels `style`-Tag und DOM-Referenzen
  - im jeweiligen HTML-Tag mittels `style`-Attribut
- alle Browser haben ein Standard-Stylesheet, welches verwendet wird, wenn keine CSS vorgegeben werden (i.d.R. weißer Hintergrund, schwarzer Text, Schriftgröße des Systems, unterstrichene Links, etc.)

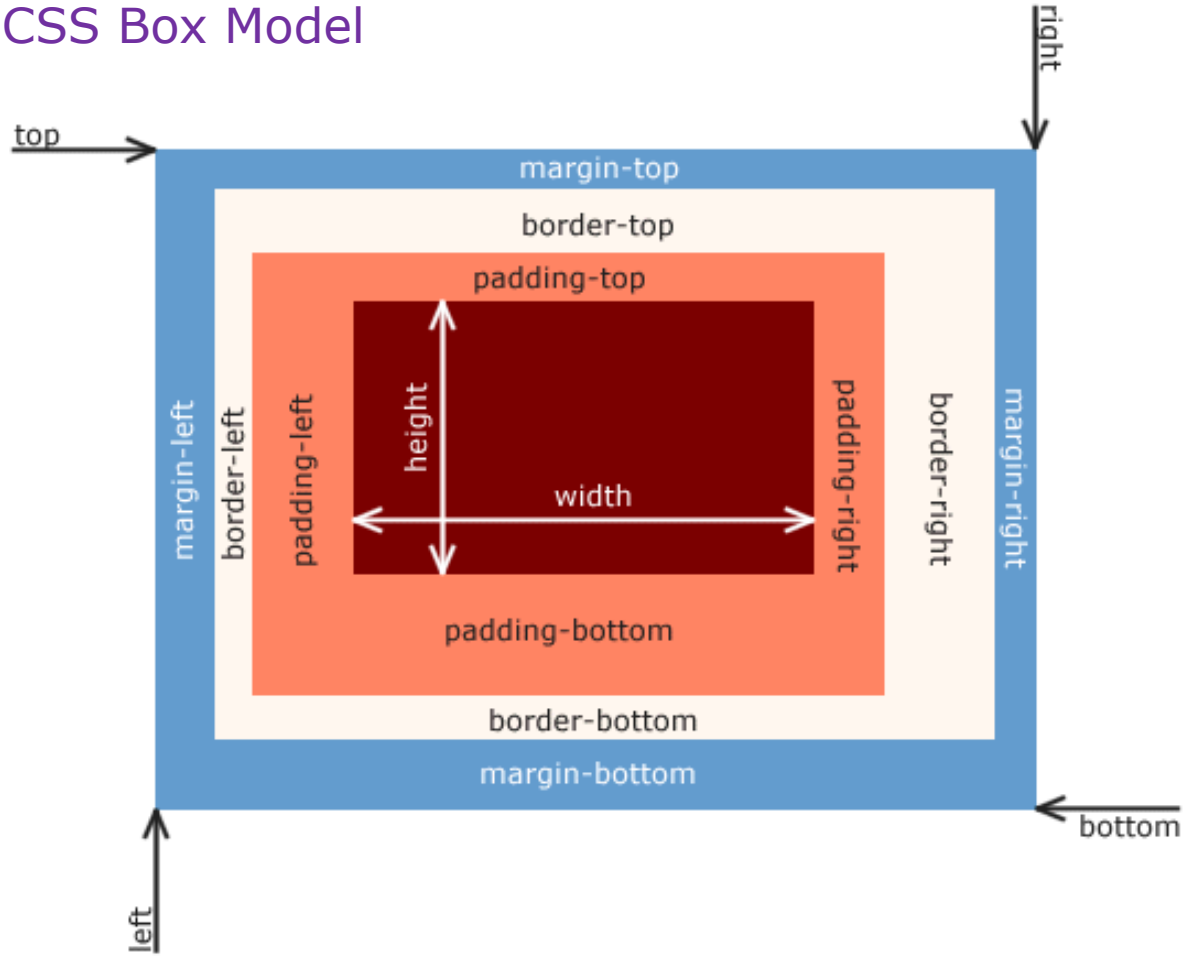
SCHRIFTLINIEN





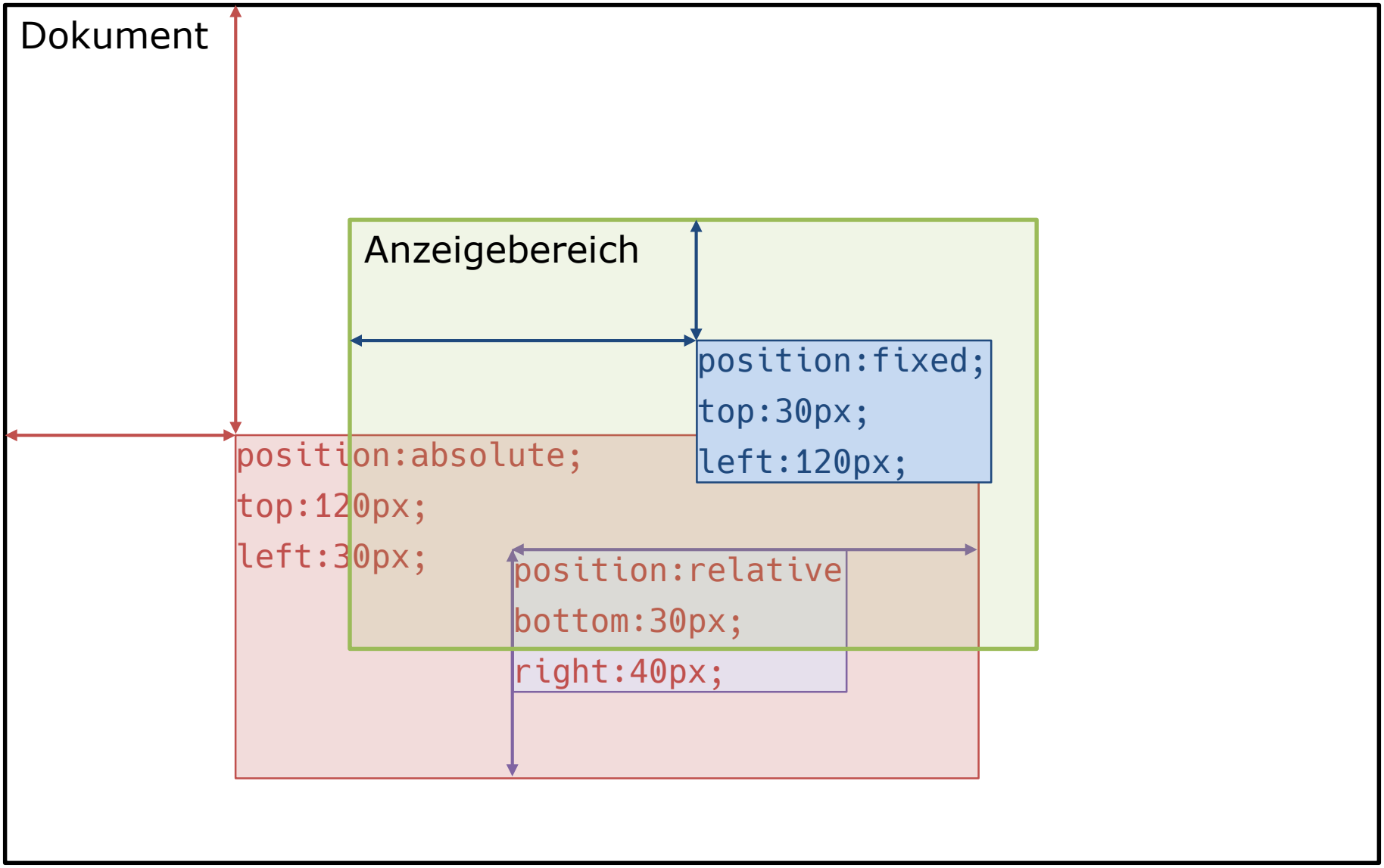
Positionierung (1/3)

CSS Box Model

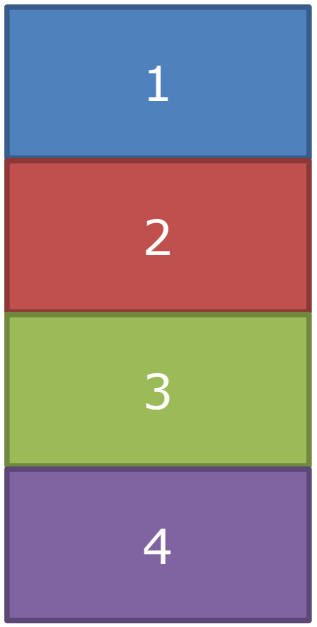


Public Domain, via Wikimedia

Positionierung (2/3)



Positionierung (3/3)




```
/* Standardverhalten
 * entspricht: */
.flt{
    float: none;
    clear: both;
}
.clr{
    clear: right;
}
/* Kann also weg
 * gelassen werden */
```


```
/* rechtsseitig umfließen */
.flt{
    float: right;
}
.clr{
    clear: right;
}
```


```
/* linkssseitig umfließen */
.flt{
    float: left;
}
.clr{
    clear: left;
}
```

```
<!-- HTML-Quelle -->
<div class="box1 flt">1</div>
<div class="box2 flt">2</div>
<div class="box3 flt">3</div>
<div class="box4 clr">4</div>
```

```
/* allgemeines CSS */
.box1{background-color:blue;border:dark-blue solid 2pt;}
.box2{background-color:red;border:dark-red solid 2pt;}
.box3{background-color:green;border:dark-green solid 2pt;}
.box4{background-color:purple;border:dark-purple solid 2pt;}
.box1,.box2,.box3,.box4{color:white;}
```

Medien

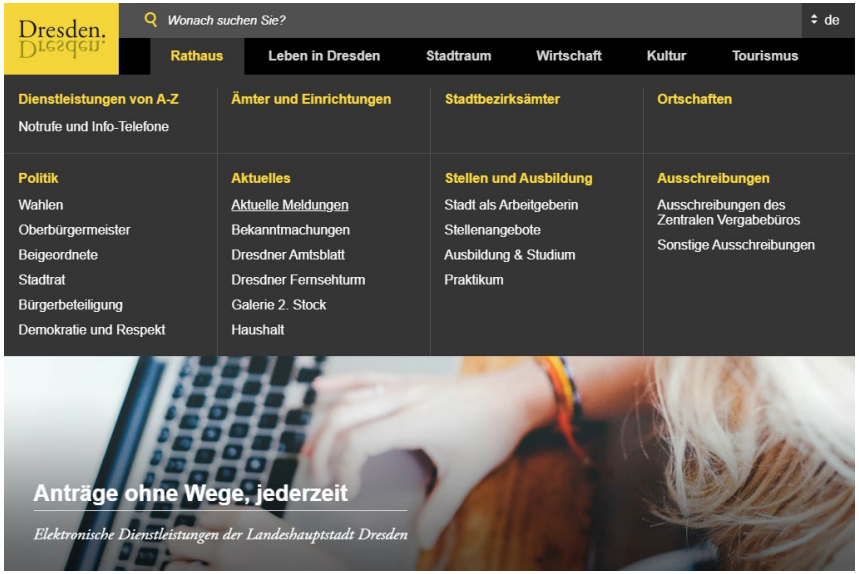
- CSS kann unterschiedliche Zielmedien definieren
  - `all` Stylesheet gilt für alle Ausgabegeräte
  - `speech/aural` Screenreader
  - `braille` Blindenschriftausgabegeräte
  - `embossed` Blindenschrift-Drucker
  - `handheld` Handhelds
  - `print` Drucker
  - `projection` Beamer
  - `screen` Bildschirme
  - `tty` Fernschreiber und Terminals
  - `tv` Fernseher
- mehrere Medientypen können in einem CSS definiert werden und sich bei Bedarf überschreiben
- Definition der einzelnen Medientypen mittels `@media` im CSS oder `media`-Attribut im `link`- oder `style`-Tag

Hypertext-Navigation

NAVIGATIONSTECHNIKEN

- für jede Webseite muss ein Navigationskonzept erstellt werden
- „Lost in (Hyper)Space"-Problem soll vermieden werden
  → Keine Sackgassen!
- Benutzer soll vorwärts und rückwärts durch die Liste bisher besuchter Seiten wandern können
- Verwenden von bekannten Metaphern
- Leitfragen von Nivergelt & Nielsen zu Plänen der Benutzenden:
  - Wo bin ich? Was kann ich hier tun?
  - Wo war ich vorher? Wie komme ich dorthin zurück?
  - Wohin kann ich gehen? Wie komme ich dorthin?

NAVIGATIONSKOMPLEXITÄT



Formulare

- Auswahllisten, Suche...

Dynamische Menüs

Bäume

Inhaltsverzeichnis

Index/Glossar

Navigationstechnik – Karten

- Karten (Image Maps) enthalten an bestimmten Region (area) Verweise
- verschiedene Regionen führen zu verschiedenen URIs
- verschiedene Formen für Regionen können verwendet werden
- Datenstruktur für das Bedienelement

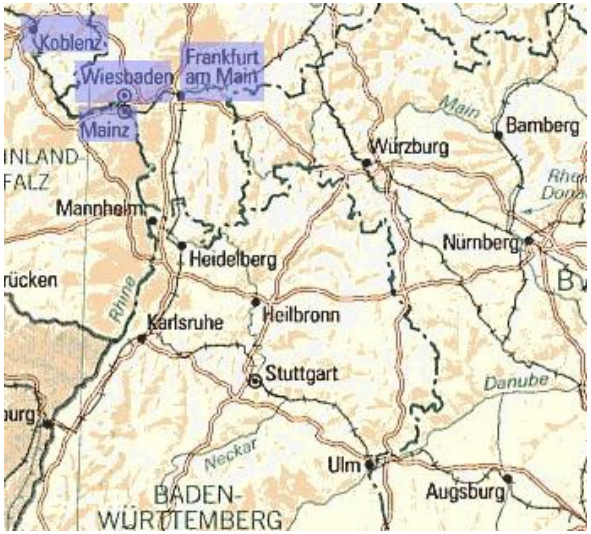
```
<map name="mapName">
<area xx="yy">
```

  - shape
  - coords
  - href
  - alt

- Verwenden der Karte: `<img usemap="#mapName" …>`

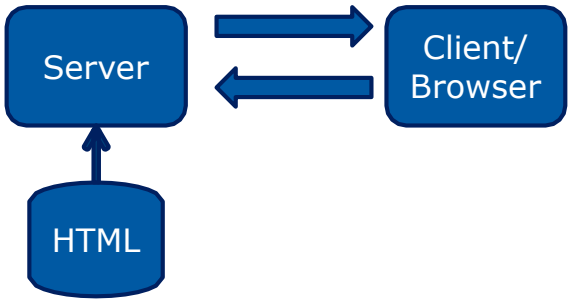
Navigationstechnik: Karten – Beispiel

```
<map name="mapOfGermany">
   <area shape="rect" coords="11,10,59,29"
      href="https://www.koblenz.de/"
      alt="Koblenz" />
   <area shape="rect" coords="42,36,96,57"
      href="https://www.wiesbaden.de/"
      alt="Wiesbaden" />
   <area shape="rect" coords="42,59,78,80"
      href="https://www.mainz.de/"
      alt="Mainz" />
   <area shape="rect" coords="100,26,152,58"
      href="https://www.frankfurt.de/"
      alt="Frankfurt" />
</map>
<p>
   <img src="karte.gif" width="345" height="312"
      border="0" alt="Karte" usemap="#mapOfGermany" />
</p>
```

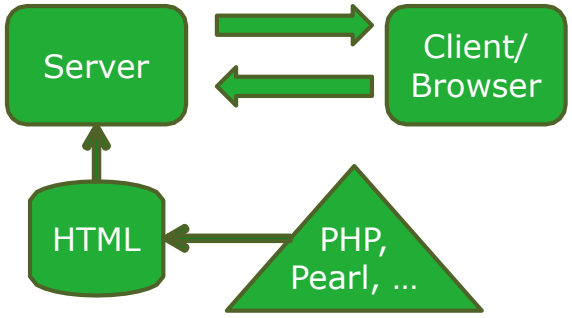


Statisches vs. Dynamisches Web

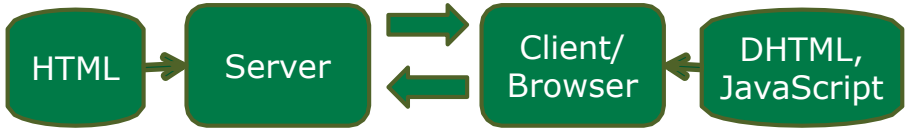
statisches Web
(Webserver liefert HTML aus)



dynamisches Web, serverseitig
(Webserver startet Programm,
das HTML generiert)



dynamisches Web, clientseitig
(Browser wertet Anweisungen
aus und verändert HTML)



JavaScript (JS)

JavaScript

- eigentlich Programmiersprache zur Erweiterung von HTML und CSS um
  - Interaktivität auf Basis von Nutzeraktionen
  - Dynamik ohne Nutzeraktionen
- heute auch serverseitig und in Microcontrollern verbreitet
- hat trotz des Namens nichts mit Java zu tun!
  (ist in ECMA 262 als ECMA Script spezifiziert)
  - dynamisch spezifiziert
  - objektorientiert
  - *klassenlos*
- erlaubt prozedurales, funktionales und objektorientiertes Programmieren
  (auch vermischt!)

Anwendungsgebiete

- dynamische Manipulation des DOM
- Datenvalidierung bei Formularen
  (vor/während/nach der Kommunikation mit dem Server)
- Anzeige von Dialogfenstern
- AJAX
- Animieren von Werbebannern oder Laufschriften
- Obfuskation (bspw. Verschleierung der E-Mail-Adresse)
- Frame-Manipulation
  - mehrere Frames auf einmal wechseln
  - Seite aus dem Frameset lösen
  - Seite in Frameset einbetten
- Schreib-/Lesezugriff auf Cookies und Web Storage

SANDBOX

- Browser führen JavaScript isoliert als Single-Thread in einer Sandbox aus
  - beschränkt Zugriff auf Objekte im Browser
  - beschränkt Zugriff auf DOM von Seiten des selben Web-Servers (Same Origin Policy; SOP) zur Erschwerung von seitenübergreifenden skriptbasierten Angriffen (Cross Site Scripting; XSS)
- einzige definierte Interaktionsmöglichkeit aus Sandbox heraus: Dateidialoge (`<input type="file" />`)
- sonstige Erweiterungen von JavaScript machen Sandbox i.d.R. unsicherer (bspw. Microsoft JScript)
- Sandboxes können unterschiedlich restriktiv sein (bis hin zur Deaktivierung von JavaScript)
  → keine Garantie, dass JavaScript in allen Umgebungen gleich agiert!

EIGENE OBJEKTTYPEN UND SICHERBARKEIT

- eigene Objekttypen werden über Konstruktorfunktionen definiert
- es gibt in JavaScript keine privaten Eigenschaften
  → stattdessen Funktionsobfuskation

```
var Cat = function () {
    var lifes = 7;
    var mew = function () {
        return (lifes > 0) ? "mew" : "argh";
    };
    return {
        kill: function () {
            lifes -= 1;
            alert(mew());
        }
    };
};
var kitty = Cat();
kitty.kill();
```

namenlose Funktion obfuskiert alle innen deklarierten Variablen und Funktionen

vom Effekt her unsichtbare Variablen und Funktionen (entspricht Sichtbarkeit „private“)

wegen return vom Effekt her öffentliche Methode (entspricht Sichtbarkeit „public“)

Fehlerbehandlung

- Fehler können aus der Laufzeit selbst eskalieren
- alternativ kann man Fehler selbst eskalieren („werfen“)

```
throw new Error("Fehlernachricht");
```

- Fehler werden mit `try…catch`-Mechanismus behandelt
- unbehandelte Fehler führen zum (sofortigen) Execution Stop

Datenpersistierung, Sitzungen

PERSISTIEREN

- ohne Persistieren der Daten gehen sie nach Schließen des HTML-Dokumentes verloren
- sitzungsübergreifende Informationen
  - Login-Status
  - Nutzerrechte
  - Voreinstellungen
  - Aktivitätshistorie
  - …

Cookies

- Erzeugung auf Server oder Client
  - Server: Übertragung im HTTP-Header an den Client
  - Client: durch JavaScript oder weitere Skriptsprachen
    (das Skript ist i.d.R. in der vom Server übermittelten Webseite)
- werden lokal auf dem Client gespeichert
- vom Browser verwaltet
- besonders zu schützen (XSS, CSRF, …)
- können mit Requests als Datagramm an den Server gesendet werden
  → ermöglicht unterbrechbare, fortsetzbare Sitzungen

AUFBAU

- als zusätzliche Zeile im HTTP-Header
- Setzen:

```
"Set-Cookie:" <Name> "=" <Wert> { ";" <Attribut>}
```

- Verwenden:

```
"Cookie:" <Name> "=" <Wert> { ";" <Name> "=" <Wert>}
```

- erlaubter Zeichenvorrat entsprechend RFC-2616

Beispiel: ACL-Cookie

Server sendet signiertes Cookie mit ACL-Rechten:

```
HTTP/1.1 200 OK↵
Set-Cookie: sessionID=P3X112llIGF1ZmJhdQ==;
   expires=Sun, 2-Jun-2019 21:01:17 GMT;
   acl=/gallery/hara/|rwx,/gallery/|r-x,/shared/|rwx,*|---;
   signature=79dbeb10d9314e83853e54f2ccebafceabb6edfa1f836a4676
   02d171978cda5868e496ab469618637e3d872b562a2ced08cb66db2b5f3e
   a8356ec2112dddf30d↵
```

Zugriff auf Cookies

- Cookies lesen:

```
allCookies = document.cookie;
```

Achtung! Liefert Semikolon-separierten String mit allen Cookies!

Zugriff auf ein bestimmtes Cookie bspw. so:

```
function getCookie(name) {
  var v = document.cookie.match(
            '(^|;) ?' + name + '=([^;]*)(;|$)');
  return v ? v[2] : null;
}
```

- Cookies schreiben:

```
document.cookie = "name=wert";
```

- Cookies löschen:

```
document.cookie = "name=";
```

JavaScript und HTML

- JavaScript-Kode kann überall im HTML-Dokument eingebunden werden
  - inline im Header oder Body

```
<script type="text/javascript">
   alert("Hallo Welt!");
</script>
```

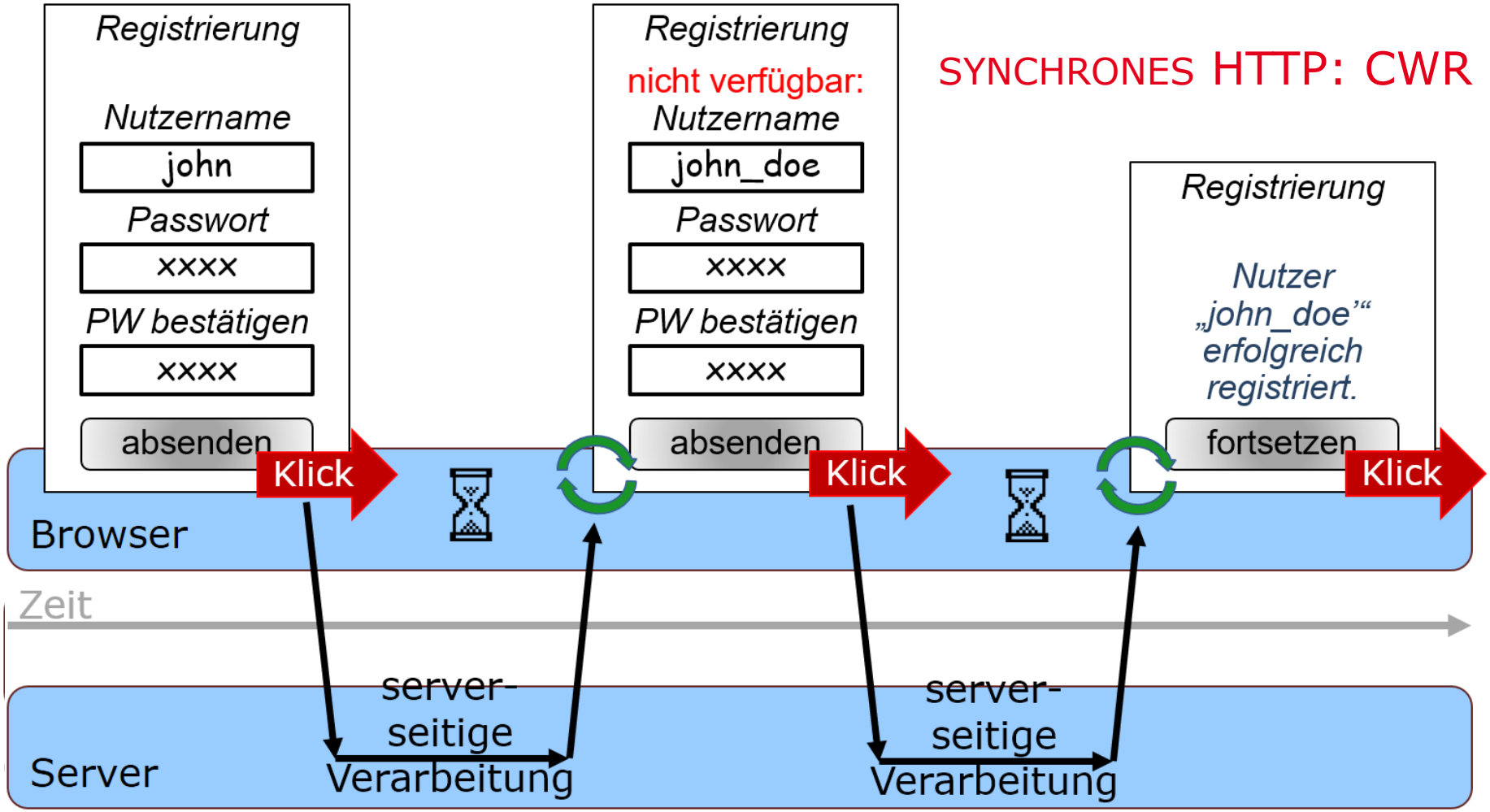
  - extern im Header oder Body

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

- JavaScript-Kode wird ausgeführt
  - im Header: sofort nachdem der HTML-Quellkode vollständig ausgewertet wurde
  - im Body: sofort nachdem die entsprechende HTML-Quellkode-Stelle ausgewertet wurde
  - nacheinander in der Reihenfolge wie der JavaScript-Kode im HTML-Quellkode steht

AJAX

SYNCHRONES HTTP: CWR



- **Click** = Nutzeraktion
- **Wait** = HTTP-Anfrage an WWW-Server → Verarbeitung der Anfrage
  → HTTP-Antwort mit vollständigem (X)HTML-Dokument an Browser
- **Refresh** = Browser aktualisiert Anzeige und zeigt neue (X)HTML-Seite

SYNCHRONES HTTP

- „synchron“ im Sinne von Nutzeraktionen
  - um Anzeige zu aktualisieren muss HTTP-Anfrage gesendet werden
  - Nutzeraktionen werden *blockiert* bis HTTP-Antwort empfangen
- hohe Korrelation zwischen HTTP-Anfrage und -Antwort

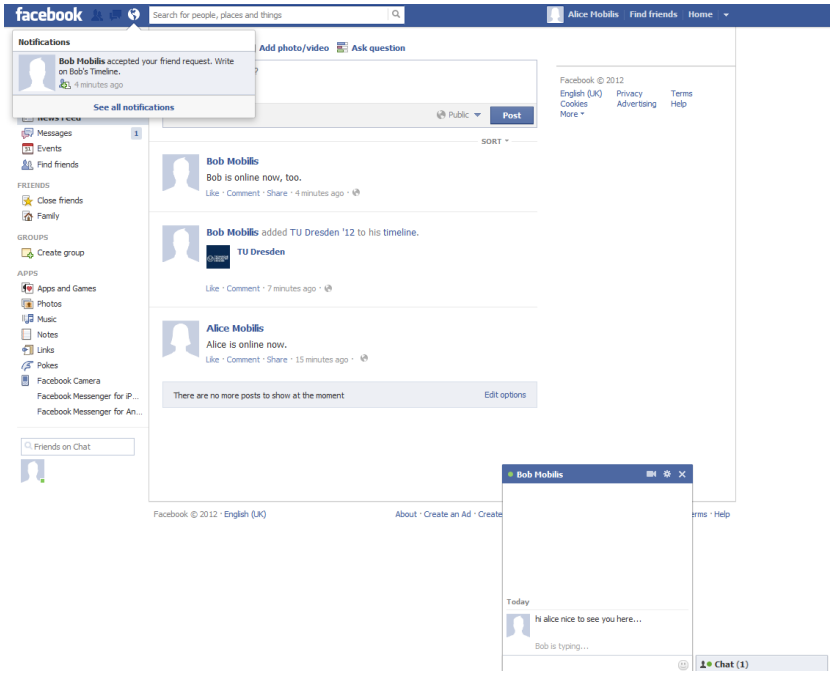
⇒ Kopplung von Nutzeraktivität an serverseitige Verarbeitung

- Nachteile des klassischen Web-Anwendung-Modells
  - schlechte Performanz wegen „Click, Wait and Refresh“ (CWR)
  - Verlust des operative Kontextes während der Seitenaktualisierung
  - exzessive Server-Last und Bandbreitenverbrauch wegen redundanter Seitenaktualisierungen

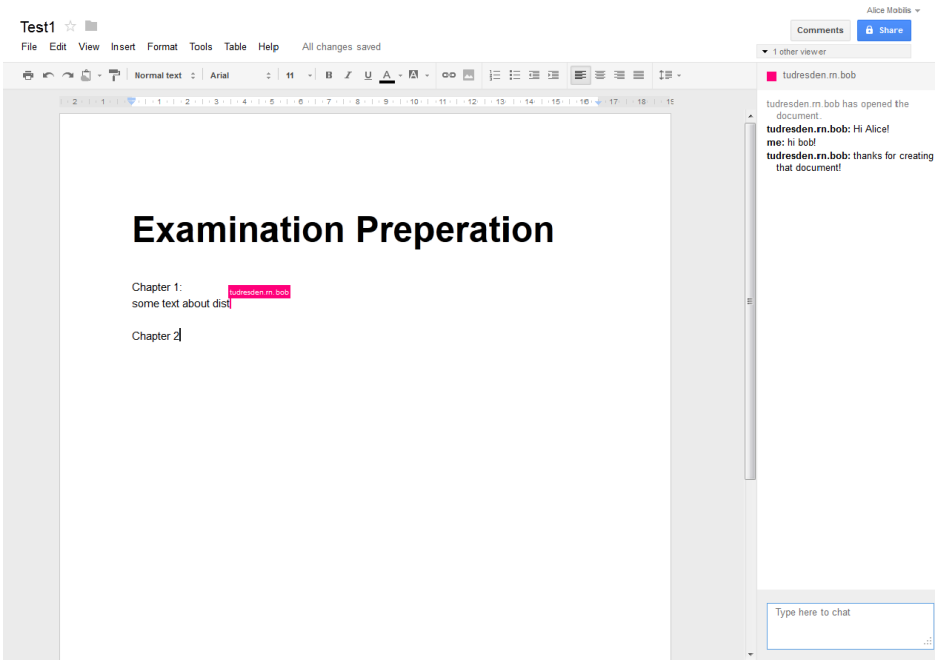
⇒ Ergebnis: langsame, unzuverlässige, vermindert produktive, wenig interaktive und ineffiziente Web-Anwendungen

AJAX: NICHT BLOCKIERENDES KOMMUNIKATIONSMODELL

Facebook



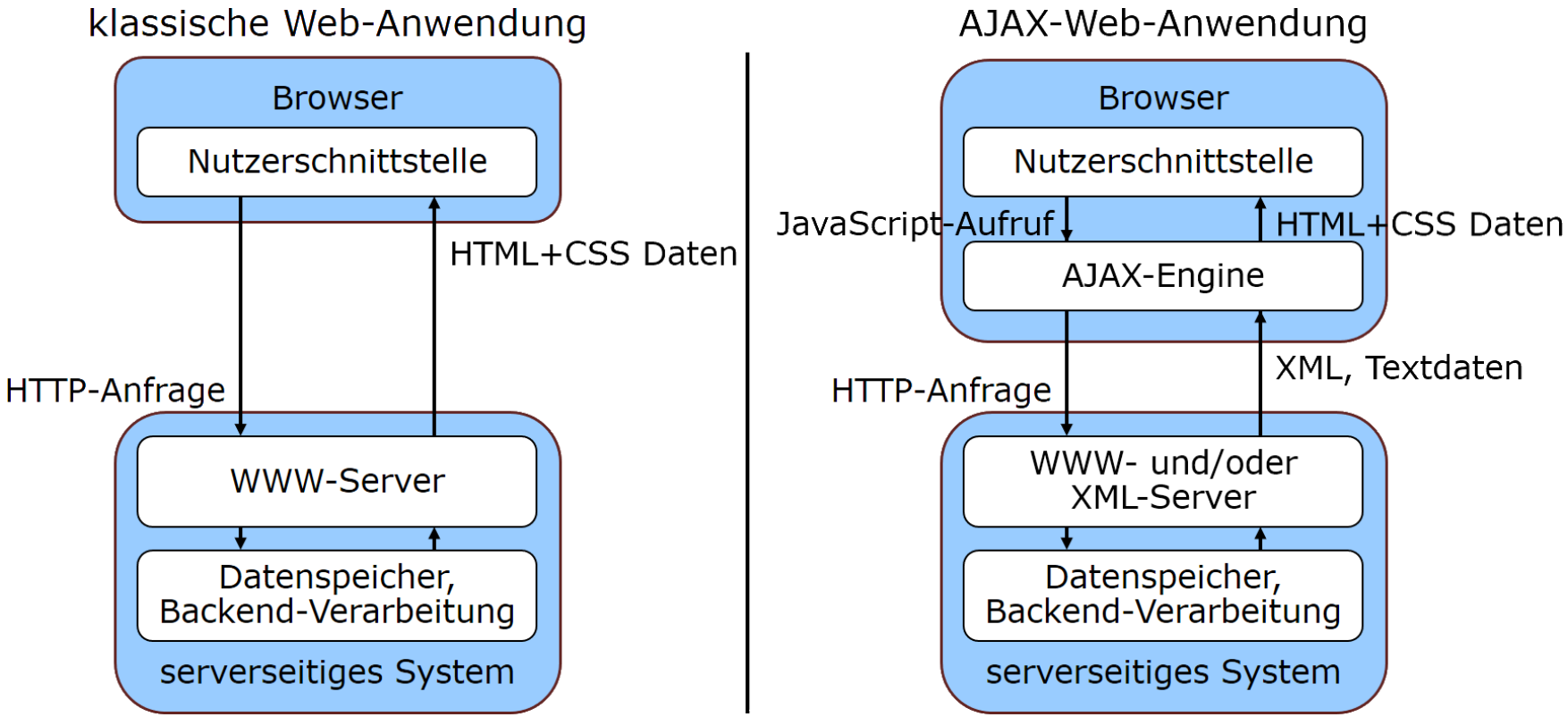
Google Docs



Intention von AJAX

- Web-Anwendungen die wie Desktop-Anwendungen aussehen und agieren
- erweitern klassischer Web-Anwendungen um Funktionalität wie dynamische Menüs, Lazy Evaluation, automatische Vervollständigung, …

DAS AJAX KONZEPT

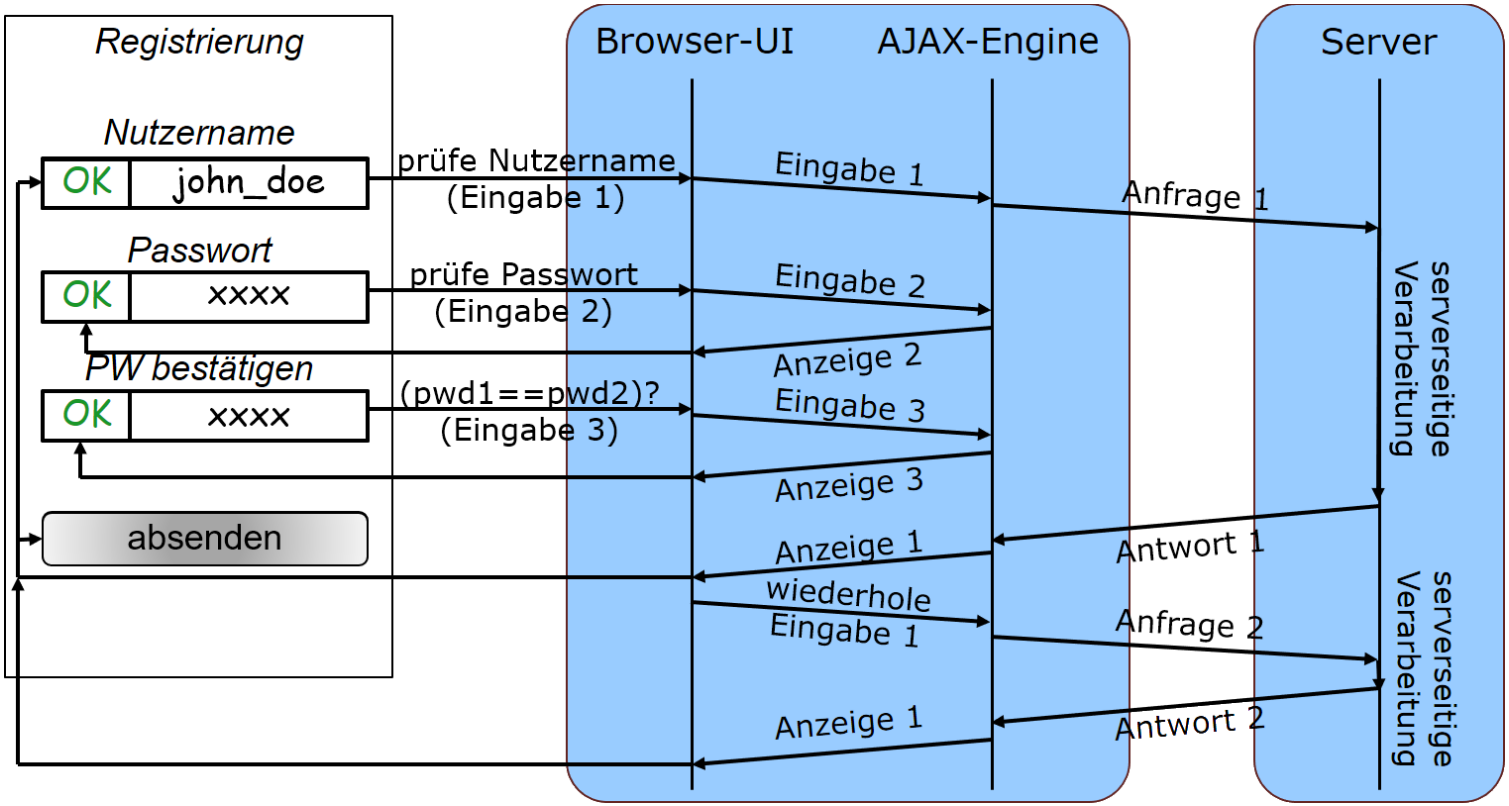


Einführen einer Indirektionsstufe — die AJAX-Engine

- eliminiert das CWR-Interaktionsparadigma
- verantwortet Rendern der GUI und Kommunikation mit dem Server

⇒ Nutzeraktionen werden von Server-Kommunikation entkoppelt

AJAX: NICHT BLOCKIERENDES MODELL



- „asynchron" bzgl. Nutzeraktionen: zur Aktualisierung der angezeigten Informationen werden Nutzeraktionen nicht durch Server-Anfragen blockiert
- hohe Abhängigkeit zwischen HTTP-Anfrage und -Antwort: jede serverseitig verarbeitete Aktion erzwingt eine RTT zwischen AJAX-Engine und WWW-Server
- Nutzeraktivität und serverseitige Verarbeitung werden entkoppelt

LITERATUR

Balzert: *Basiswissen Web-Programmierung. XHTML, CSS, JavaScript, XML, PHP, JSP, ASP.NET, AJAX, W3L*
Verlag Herdecke, aktuelle Auflage

http://www.w3schools.com/

http://www.w3fools.com/

http://de.selfhtml.org/html/index.htm

http://de.selfhtml.org/css/index.htm

http://www.knowware.de/home_html_css.html