

WBE: WEB-ENTWICKLUNG

EINFÜHRUNG

ÜBERSICHT

- Worum geht's in WBE?
- Organisatorisches
- Internet und WWW
- Client-Server-Architektur
- Die Sprachen des Web: HTML, CSS, JavaScript
- Web-Standards und APIs

ÜBERSICHT

- Worum geht's in WBE?
- Organisatorisches
- Internet und WWW
- Client-Server-Architektur
- Die Sprachen des Web: HTML, CSS, JavaScript
- Web-Standards und APIs

WORUM GEHT'S?

- Web-Technologien...?
 - HTML, CSS, JavaScript
 - Canvas, SVG, Geolocation, DOM, Websockets, uvm.
- Web-Plattform...?
 - Web-APIs und deren Einsatzmöglichkeiten
 - z.B. Web-Apps, Electron-Apps, Webserver uvm.
- Web-Entwicklung...?
 - Tools, Frameworks, Libraries

Das ist ein riesiges Gebiet!

ZIELE WBE

- JavaScript gut verstehen und einsetzen können
- Nebenbei: HTML und CSS verstehen (Selbststudium)
- Auswahl von Web-APIs kennen und einsetzen können
- Eigenes kleines Web-Framework erstellen können

Klar ist: Das Vorhaben ist immer noch sehr ambitioniert

ERWARTETE VORKENNTNISSE

- Grundkenntnisse HTML und CSS
 - Mail am Ende des letzten Semesters
 - Moodle-Kurs zum Selbststudium
 - In Tests und Praktikumsaufgaben vorausgesetzt
- Grundlegende Programmierkenntnisse
 - Programmieren 1/2
 - Software-Projekt 1/2

ÜBERSICHT

- Worum geht's in WBE?
- Organisatorisches
- Internet und WWW
- Client-Server-Architektur
- Die Sprachen des Web: HTML, CSS, JavaScript
- Web-Standards und APIs

ABLAUF

- Zwei Unterrichtslektionen pro Woche
- Zwei Praktikumslektionen pro Woche

Zeit	Art
100..120h	ca. Arbeitsaufwand 4 Credits (25..30h pro Credit)
42h	Präsenzlektionen ($14 * 4 * 0.75$)
58..78h	Selbststudium, Prüfungsvorbereitung, Prüfung

KURSNOTE

Anteil	Art
20%	Leistungsnachweise im Semester
80%	Semesterendprüfung (SEP)

LEISTUNGSNACHWEIS IM SEMESTER

- Kurztests
- Bewertete Abgaben im Praktikum

Leistungsnachweise im Semester können die Kursnote verbessern, nicht aber verschlechtern – für misslungene oder verpasste Leistungsnachweise wird die Note der SEP eingesetzt

Tipp: Nutzen Sie die Gelegenheit, sich durch die Leistungsnachweise im Semester ein Polster für die SEP zu verschaffen

SEMESTERENDPRÜFUNG

- Mix aus verschiedenen Aufgaben zu den WBE-Themen
 - Multiple-Choice (Variante Kprim)
 - Fragen mit kurzen Antworten
 - Code-Aufgaben
- *Wir haben noch keine komplette Musterprüfung* 😊
- Tests können aber als Muster für die SEP angesehen werden

UNTERLAGEN

In elektronischer Form (PDF) im Moodle:

- Die im Unterricht gezeigten Folien
- Ergänzende Hinweise zu den Folien (Lecture Notes)
- Praktikums-Aufgaben und Beispiel-Code

<https://moodle.zhaw.ch/course/view.php?id=20705>

UNTERRICHT UND PRAKTIKUM

Praktikumslektionen

- An den Notebooks gearbeitet
- Gespräche, Diskussionen, Teamarbeit erwünscht

Unterrichtslektionen

- Vortrag und Demos, Diskussionen mit ganzer Klasse
- Gespräche untereinander bitte vermeiden – führen zu schlechterem Unterricht und stören alle
- Notizen machen empfohlen

Fragen?



HINWEIS

Zum Einstieg in die WBE-Themen sind in den folgenden Abschnitten
einige Eckpunkte der Web-Grundlagen noch einmal zusammengestellt

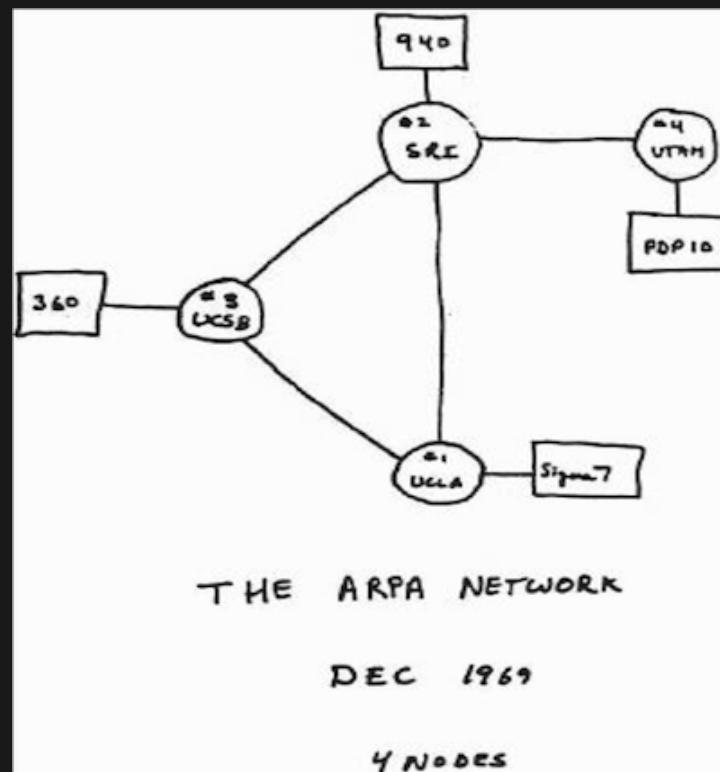
Diese sind Gegenstand des Vorbereitungskurses (Moodle), wo Sie
weiteres Material zu diesen Themen finden

ÜBERSICHT

- Worum geht's in WBE?
- Organisatorisches
- Internet und WWW
- Client-Server-Architektur
- Die Sprachen des Web: HTML, CSS, JavaScript
- Web-Standards und APIs

INTERNET

- Weltweites Netzwerk, bestehend aus vielen Rechnernetzwerken
- Ursprünglich: ARPANET (1969: vier Knoten)
- Als Internet ab 1987 bezeichnet (ca. 27 000 Knoten)



Dienste im Internet:
E-Mail, Usenet, S/FTP, World Wide Web,...

WORLD WIDE WEB

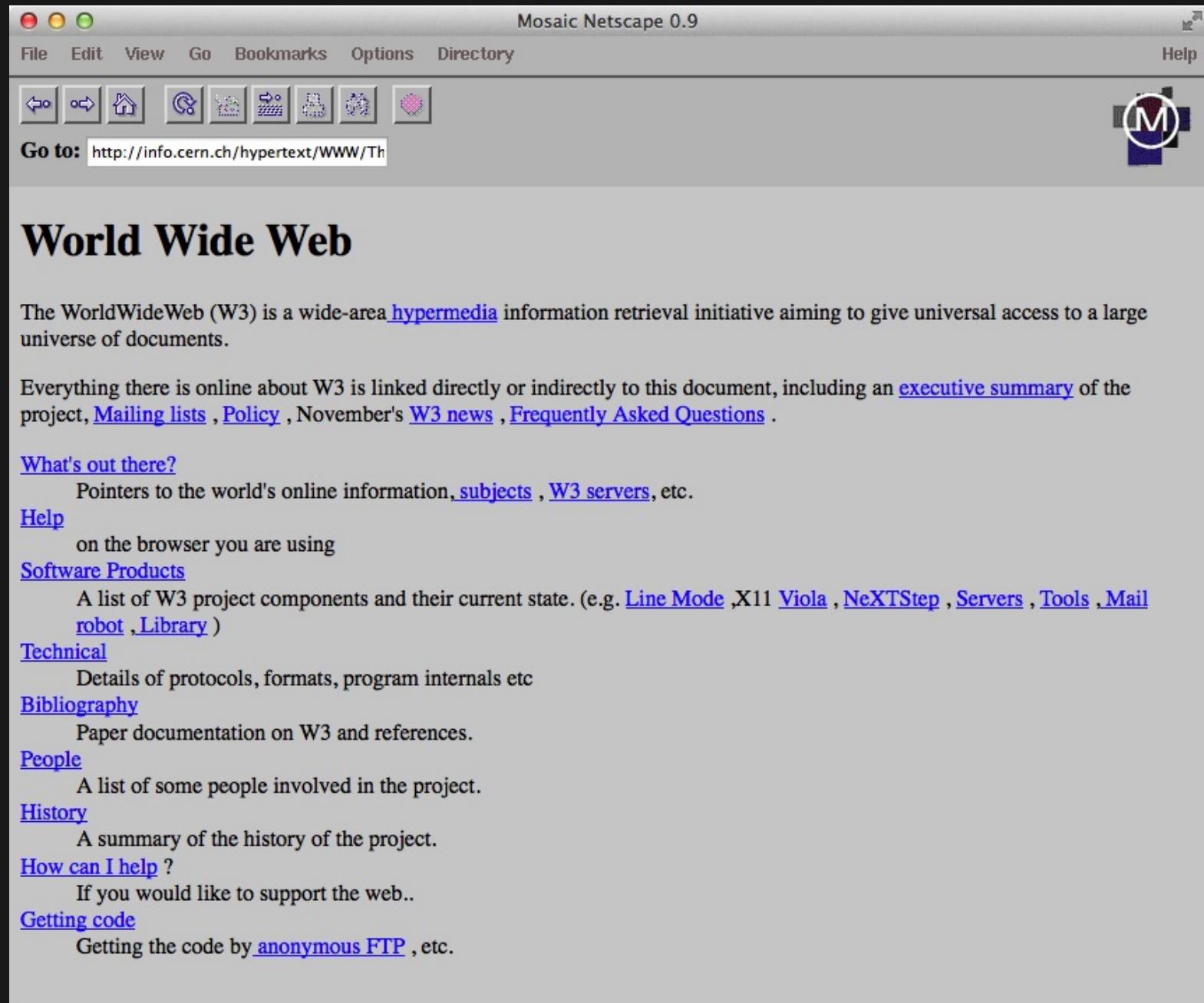
- Service, der auf dem Internet aufbaut
- Tim Berners-Lee
(Anfang der 90er Jahre am CERN in Genf auf NeXT Computer)



Tim's erster Webserver

Ankündigung des WWW im Usenet

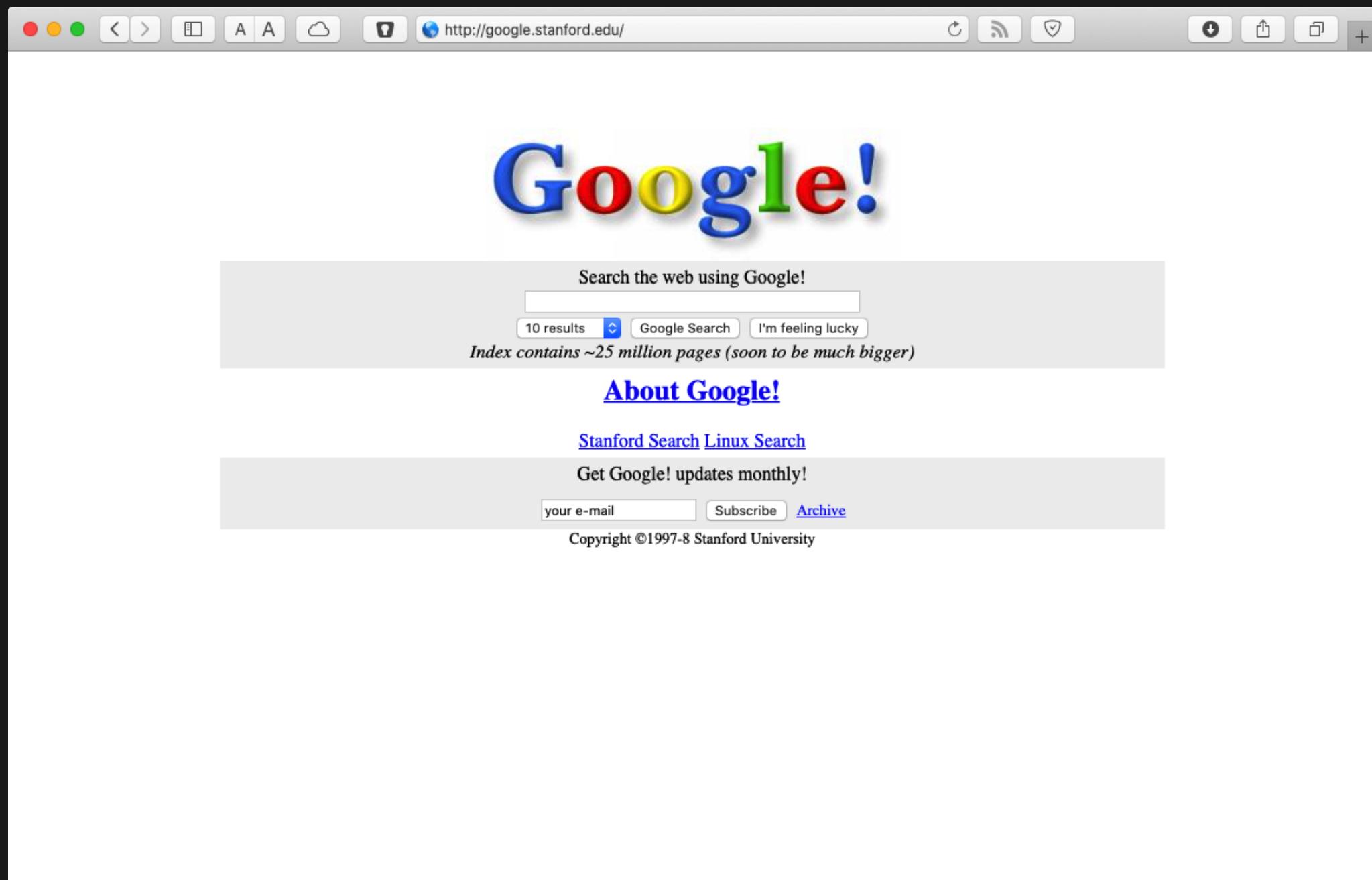
WEB-GESCHICHTE 1992



The birth of the Web
(CERN)

Ankündigung NCSA
Mosaic

WEB-GESCHICHTE 1996...



WEB HEUTE

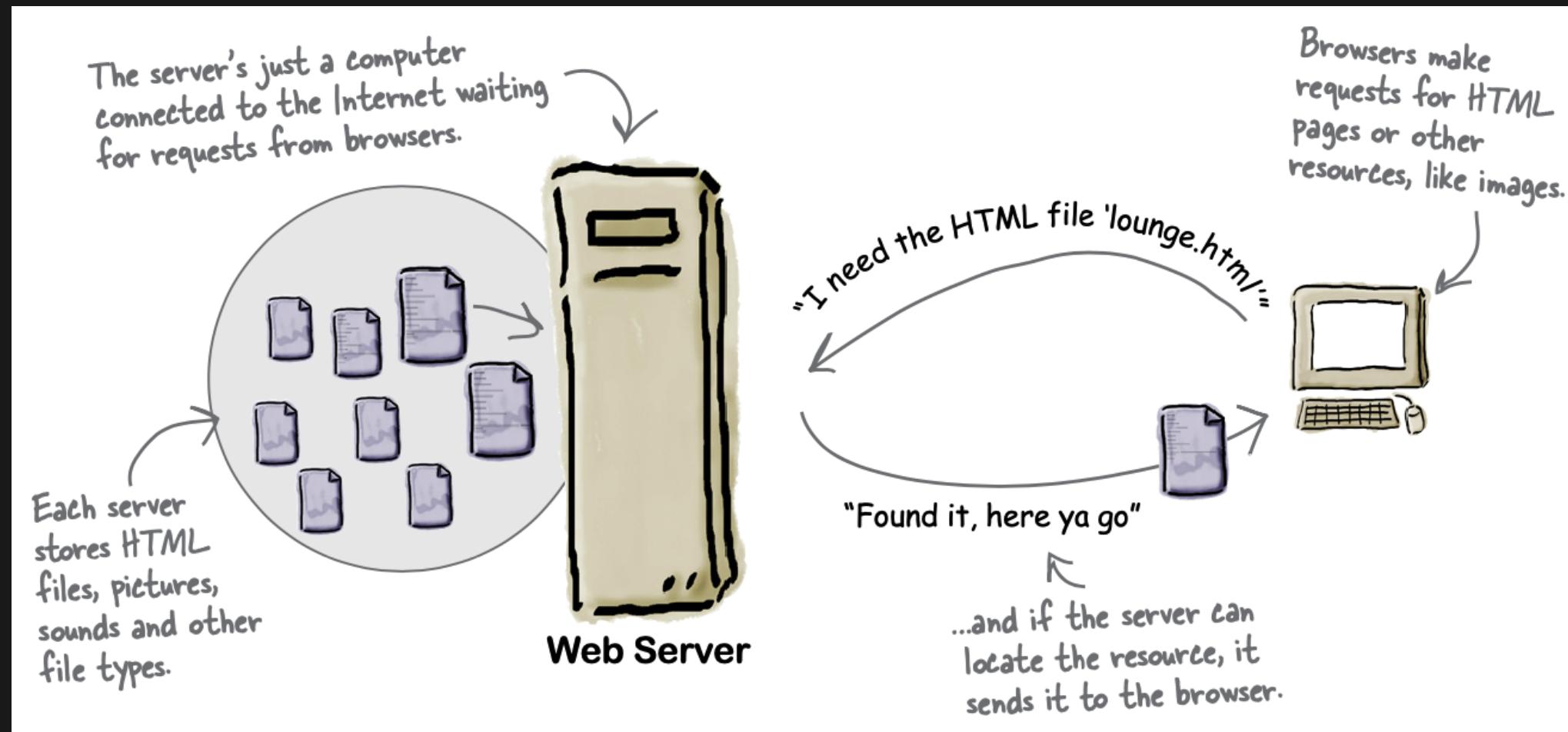
- Wichtige Applikations- und Informationsplattform
- Unzählige Technologien und Spezifikationen
- Vieles davon rund um „HTML Living Standard“
- Zahlreiche weitere APIs und Standards
- Grundlagen: HTML + CSS + JavaScript

ÜBERSICHT

- Worum geht's in WBE?
- Organisatorisches
- Internet und WWW
- Client-Server-Architektur
- Die Sprachen des Web: HTML, CSS, JavaScript
- Web-Standards und APIs

WEB-ARCHITEKTUR

- Client-Server
- Interaktion: Request/Response

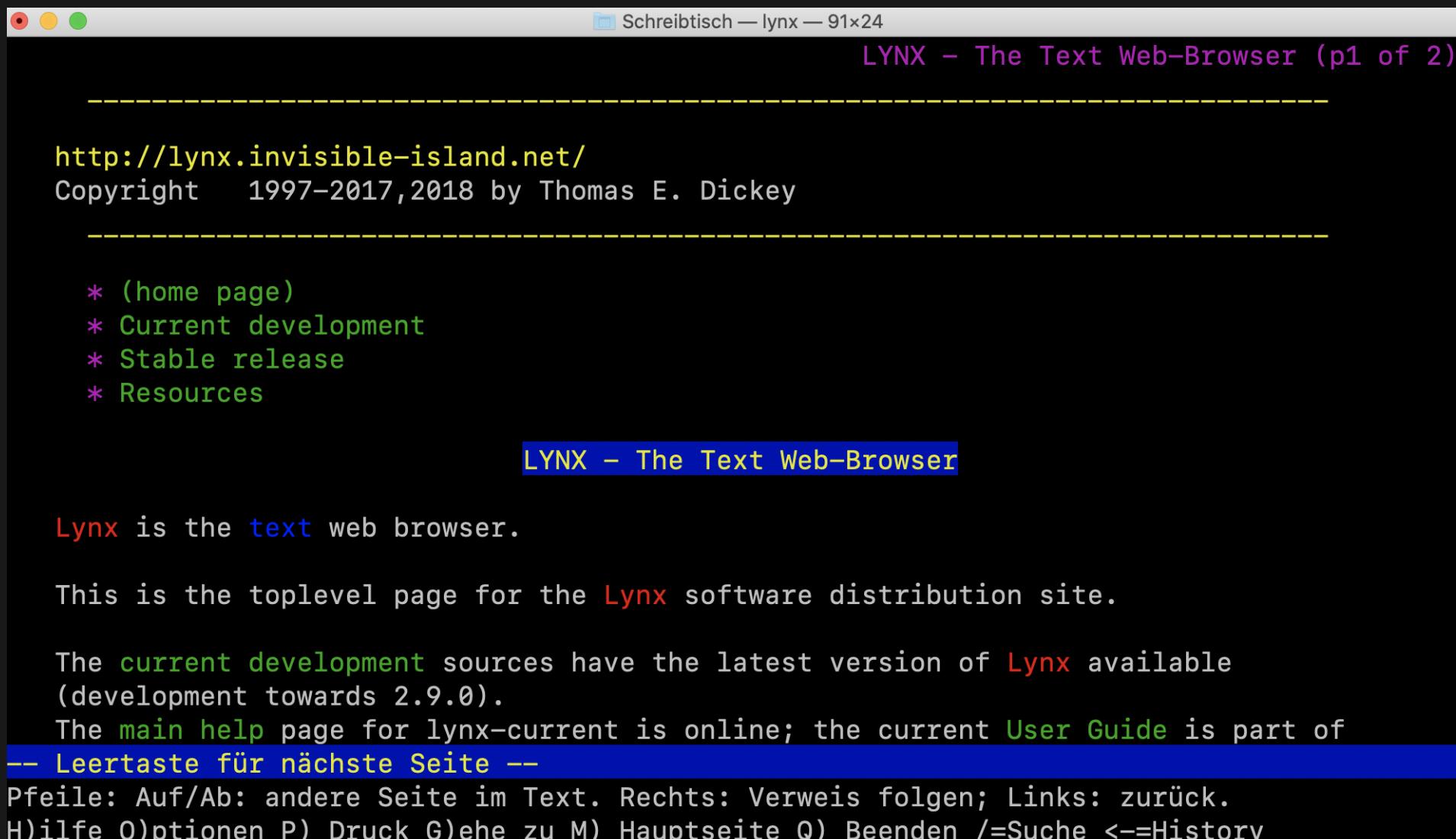


CLIENT: BROWSER / USER AGENT



<https://gs.statcounter.com>

USER AGENT: LYNX



BROWSER WARS

- Zunächst: Mosaic
- Später: Netscape
- Dann: Internet Explorer
- Heute: Chrome

Problematisch waren immer die Phasen, in denen ein Browser deutlich marktbeherrschend war

„This page is best viewed with Internet Explorer 6 on Windows.“

WEB SERVER

- Reagieren auf Anfragen von User Agents
- Senden diesen die angefragten Dateien
- Beispiele
 - Apache HTTP Server
 - Nginx (ausgesprochen „engine-ex“)
 - Microsoft Internet Information Services (IIS)
 - Apache Tomcat (Java)

WEB-PLATTFORM

Drei Grundpfeiler

- Sprachen: Rolle von HTML, CSS und JavaScript
- Adressierung: URI/URL, IP-Adresse, Domain-Name
- Protokoll: HTTP

Gleich mehr zu den Sprachen. Schauen wir zuerst kurz auf Adressierung und Protokoll...

ADRESSIERUNG

Uniform Resource Locator, URL

```
"http://" [user[:password]@] host [:port] path [ "?" query ] [ "#" fragment ]  
scheme-specific-part → → |  
|  
http://hans:geheim@example.org:80/demo/example.cgi?land=de&stadt=aa#geschichte  
| | | | | | | |  
| | | host | url-path | | query | fragment  
| | password | port | |  
| user | | | |  
scheme (hier gleich Netzwerkprotokoll)
```

(Quelle: Wikipedia)

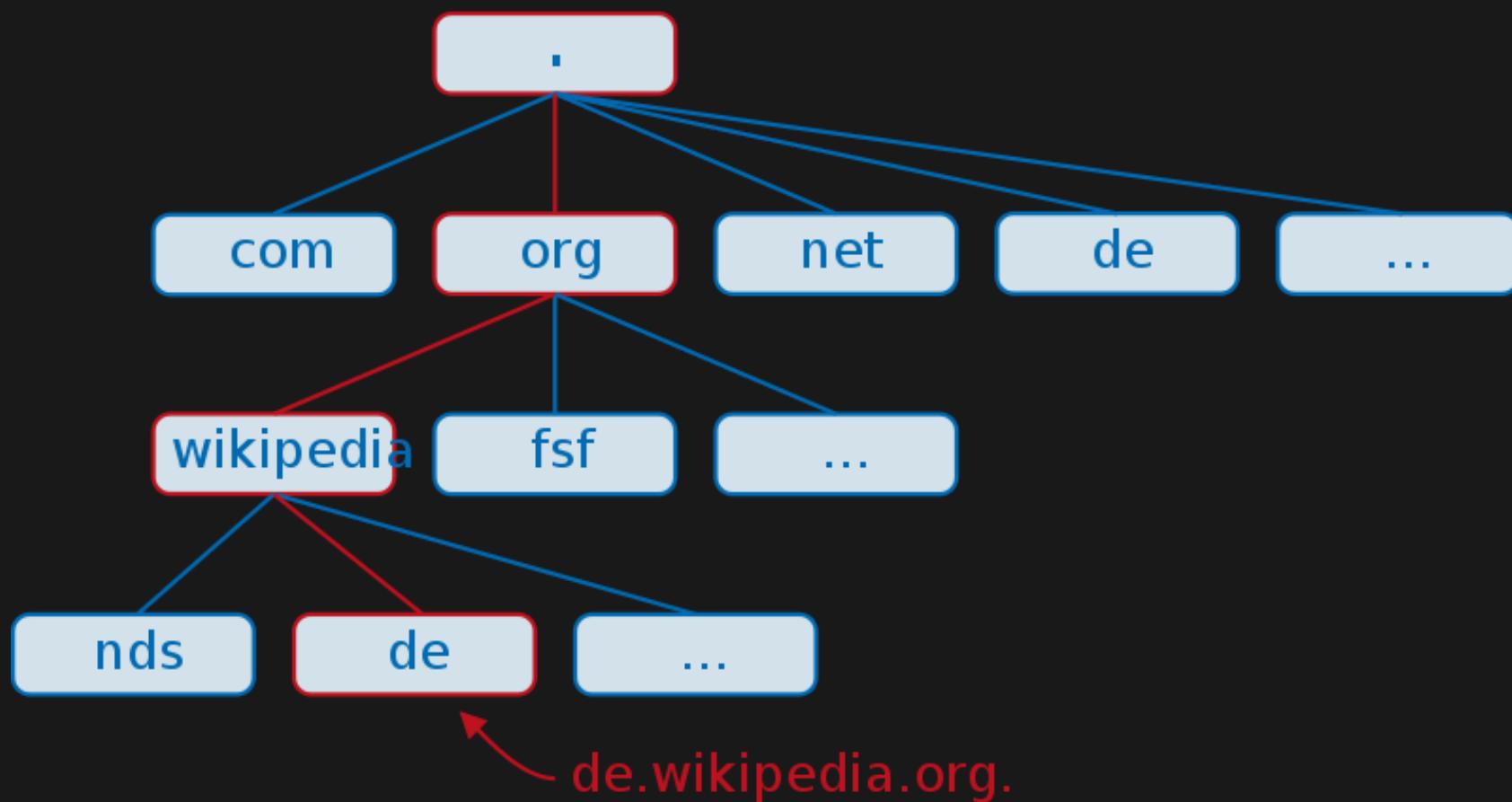
IP-ADRESSE

- Host-Angabe in URL: IP-Adresse oder Domain-Name
- IPv4: 32 Bit, IPv6: 128 Bit
- Beispiele:

173.194.40.95

243f:6a88:85a3:08d3:1319:8a2e:0370:7344

DOMAIN NAME SYSTEM (DNS)



IP-Adressen schwer
zu merken, Abhilfe:
Domain-Namen

PROTOKOLL: HTTP

- Einfaches, textbasiertes Protokoll
- Beispiel: Anfrage an Server

```
GET /~bkrt/hallo.html HTTP/1.1
```

```
Host: dublin.zhaw.ch
```

```
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X...) Gecko/20100101 Firefox/22.0
```

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
```

```
Accept-Language: de-de,de;q=0.8,en-us;q=0.5,en;q=0.3
```

```
Accept-Encoding: gzip, deflate
```

```
Connection: keep-alive
```

ÜBERSICHT

- Worum geht's in WBE?
- Organisatorisches
- Internet und WWW
- Client-Server-Architektur
- Die Sprachen des Web: HTML, CSS, JavaScript
- Web-Standards und APIs

DIE SPRACHEN DES WEB

- HTML und CSS bereits bekannt
(Vorkenntnisse, Vorkurs)
- Daher hier nur eine schnelle Zusammenfassung...

ÜBERSICHT

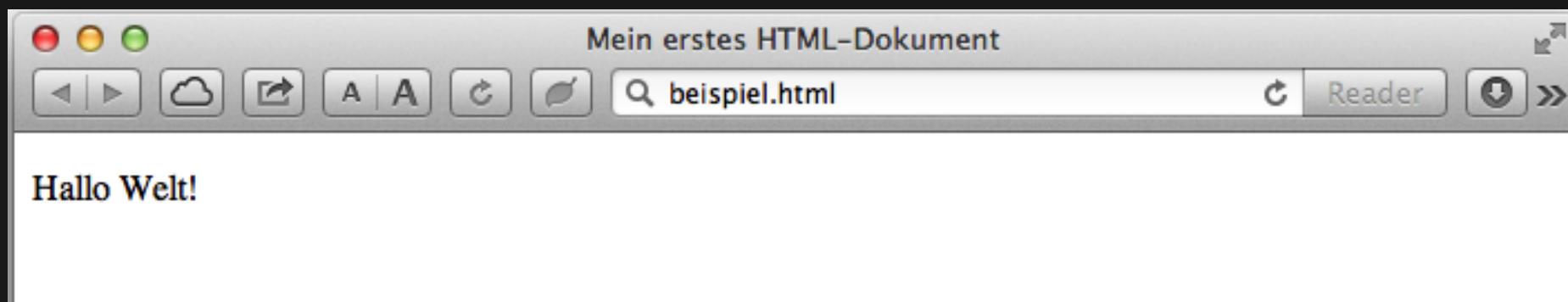
- Internet und WWW
- Client-Server-Architektur
- Die Sprachen des Web: HTML, CSS, JavaScript
 - Markup mit HTML
 - Darstellung mit CSS
 - Verhalten mit JavaScript
- Web-Standards und APIs

MARKUP MIT HTML

- Markup Language: Tags (`<p>`)
- Hypertext: Links
- Multimedia: eingebettete Multimedia-Elemente
- Gestaltung: verbundene CSS-Datei(en)
- Verhalten: verbundene JavaScript-Datei(en)
- Verschiedene Content-Types: `text/html`, `image/jpeg`, ...

MARKUP MIT HTML

```
1 <!DOCTYPE html>
2 <html lang="de">
3   <head>
4     <title>Mein erstes HTML-Dokument</title>
5     <meta charset="utf-8" />
6   </head>
7
8   <body>
9     <p>Hallo Welt!</p>
10  </body>
11 </html>
```



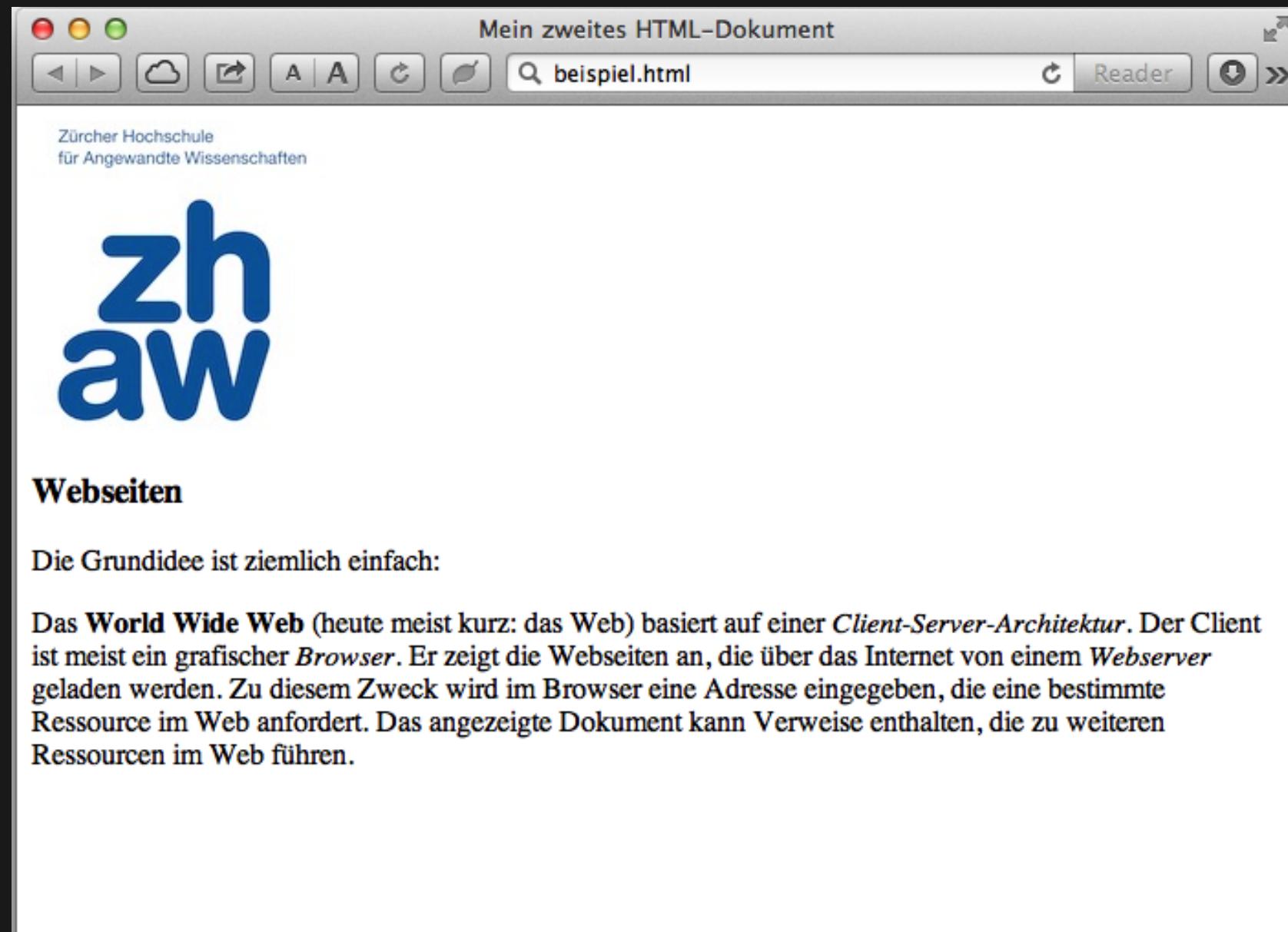
MARKUP MIT HTML

- Es geht um **Inhalt** und **Struktur** von Dokumenten
- Was bedeuten die einzelnen Teile (Überschrift? Absatz?)
- Darstellung und Verhalten explizit ausgeklammt
- Ohne CSS-Stylesheet: Standard-Darstellung im Browser

HTML: NOCH EIN BEISPIEL

```
1 <!DOCTYPE html>
2 <html lang="de">
3   <head> ... </head>
4   <body>
5     <section>
6       <header>
7         
8       </header>
9       <article>
10      <h1>Webseiten</h1>
11      <p>Die Grundidee ist ziemlich einfach:</p>
12      <p>Das <strong>World Wide Web</strong> (heute meist kurz: das Web)
13        basiert auf einer <em>Client-Server-Architektur</em>. Der Client
14        ist meist ein grafischer <em>Browser</em>. ...</p>
15      </article>
16    </section>
17  </body>
18 </html>
```

STANDARD-DARSTELLUNG (OHNE CSS)

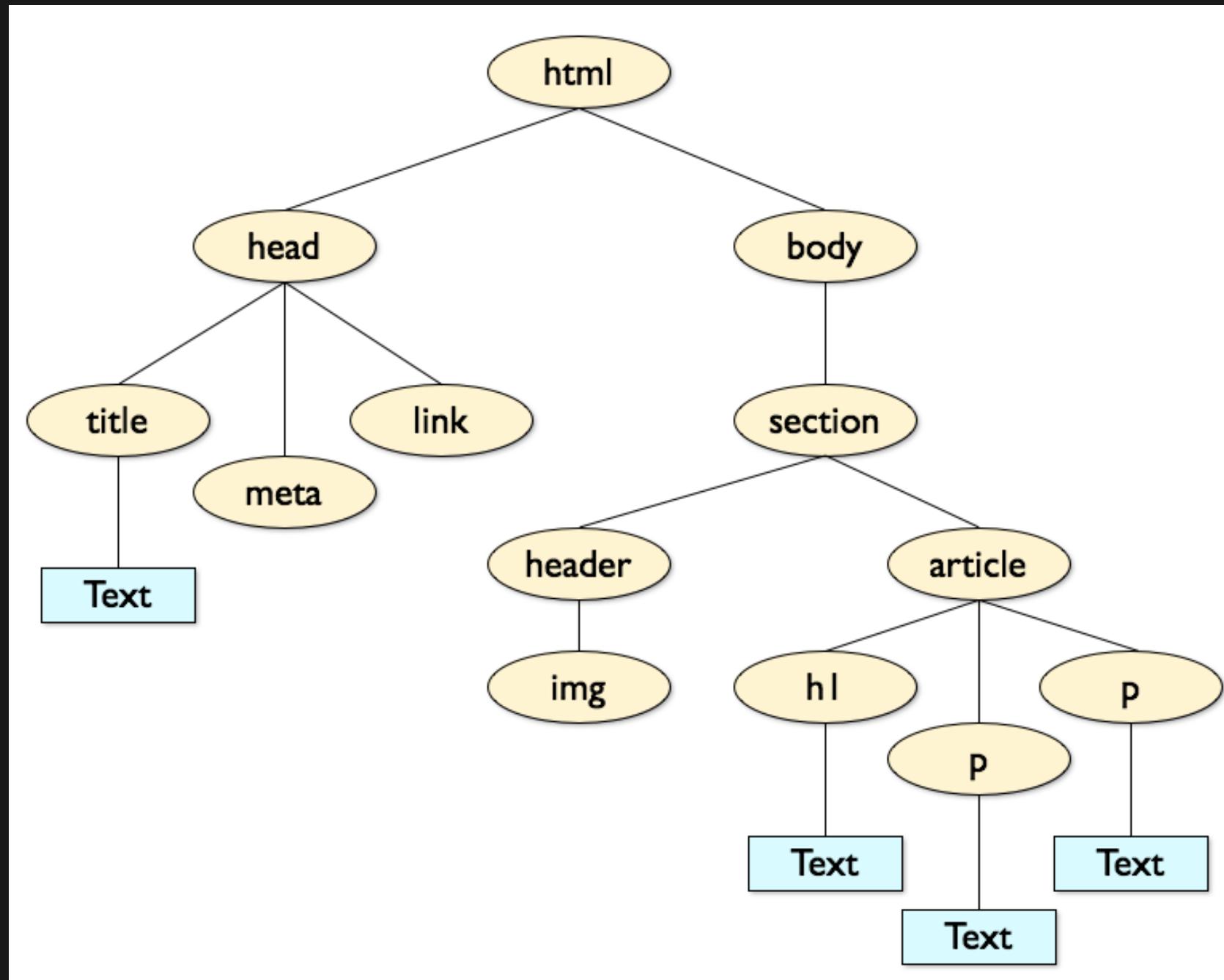


Webseiten

Die Grundidee ist ziemlich einfach:

Das **World Wide Web** (heute meist kurz: das Web) basiert auf einer *Client-Server-Architektur*. Der Client ist meist ein grafischer *Browser*. Er zeigt die Webseiten an, die über das Internet von einem *Webserver* geladen werden. Zu diesem Zweck wird im Browser eine Adresse eingegeben, die eine bestimmte Ressource im Web anfordert. Das angezeigte Dokument kann Verweise enthalten, die zu weiteren Ressourcen im Web führen.

HIERARCHISCHE STRUKTUR



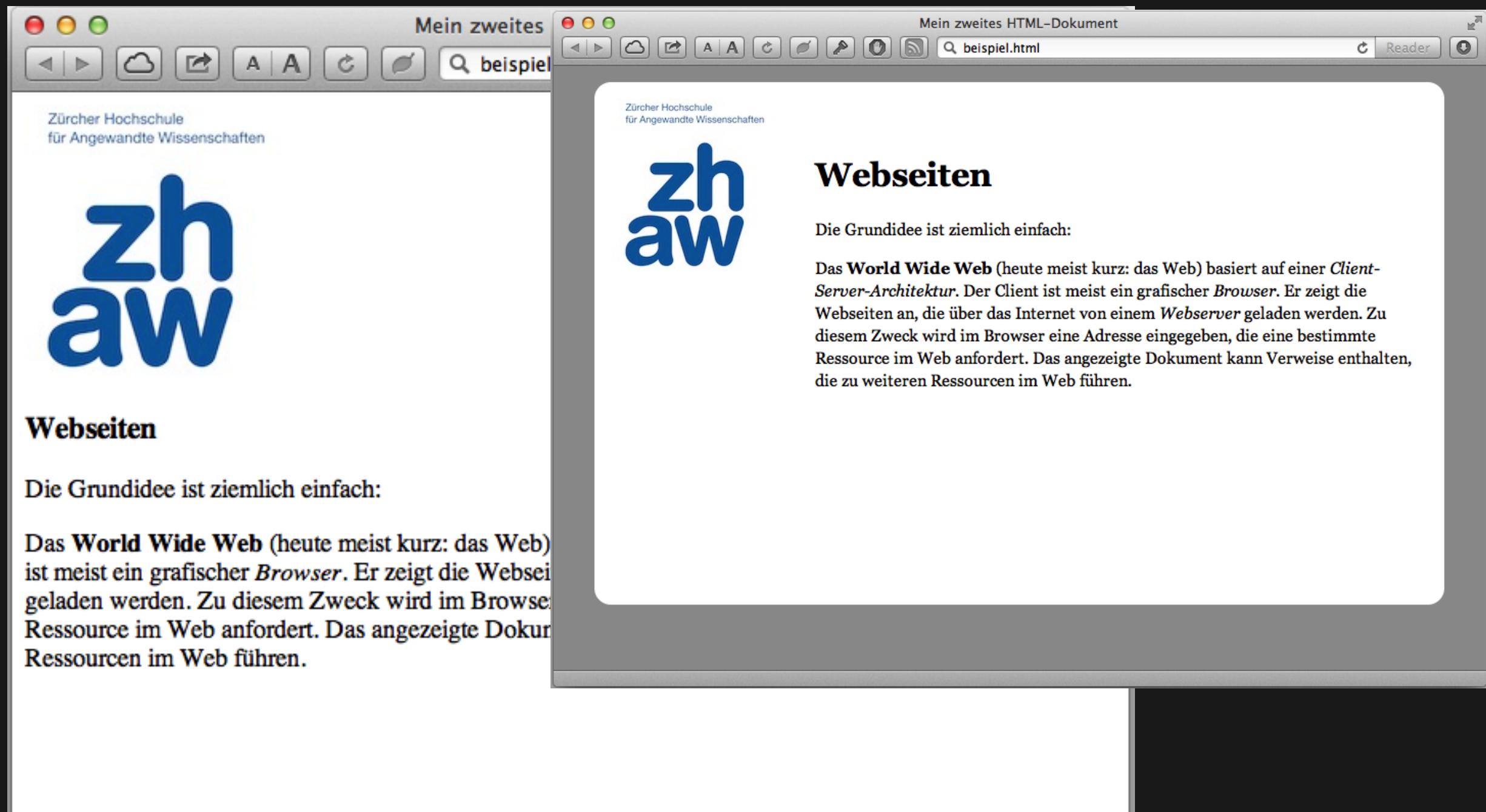
ÜBERSICHT

- Internet und WWW
- Client-Server-Architektur
- Die Sprachen des Web: HTML, CSS, JavaScript
 - Markup mit HTML
 - Darstellung mit CSS
 - Verhalten mit JavaScript
- Web-Standards und APIs

CASCADING STYLE SHEETS (CSS)

```
1 @charset "utf-8";
2
3 body {
4   background-color: #999;
5 }
6 article {
7   margin-left: 200px;
8   font-family: 'Georgia Pro', georgia, serif;
9   line-height: 140%;
10 }
11 p {
12   font-size: 3em;
13   color: blue;
14 }
15 ...
```

DARSTELLUNG OHNE/MIT CSS



ÜBERSICHT

- Internet und WWW
- Client-Server-Architektur
- Die Sprachen des Web: HTML, CSS, JavaScript
 - Markup mit HTML
 - Darstellung mit CSS
 - **Verhalten mit JavaScript**
- Web-Standards und APIs

VERHALTEN MIT JAVASCRIPT

```
1 let pElem = document.querySelector('p')
2 pElem.addEventListener('click', function () {
3   let newText = document.createTextNode("removed!")
4   pElem.replaceChild(newText, pElem.childNodes[0])
5 })
```

- Klick auf den Text ersetzt ihn durch "removed!"
- Element des Dokuments auswählen
- Reaktion auf Ereignis festlegen

JAVASCRIPT

- Veröffentlich 1995 in Vorversion des Netscape Navigators 2.0
- Unter Zeitdruck entwickelt von Brendan Eich
- Ziel: Scripts um Webseiten dynamischer zu machen
- Zunächst: LiveScript, dann JavaScript (Marketing)
- JavaScript und Java haben wenig gemeinsam (!)

JAVASCRIPT HEUTE

- Web-Apps mit vielen tausend Codezeilen (z.B. Maps)
- Einsatz in nativen Applikationen (z.B. VSCode)
- Performant: Just-in-time Compiler (JIT)
- Zielsprache für andere Sprachen (z.B. TypeScript)

JavaScript ist heute eine der wichtigsten und meistverwendeten Programmiersprachen

JAVASCRIPT PLATTFORMEN

- **Web-Browser** mit verschiedenen JavaScript-Engines und diversen APIs für die Browser-Plattform (DOM, Storage, ...)
- **Node.js** als browserunabhängige JavaScript-Plattform, ebenfalls mit speziellen APIs (Filesystem, Web-Server, ...)
- Alternativen zu Node.js wie **Deno** oder **Bun**

JAVASCRIPT-ALTERNATIVEN

- Sprachen, welche nach JavaScript übersetzt werden, u.a.
 - [TypeScript](#)
 - [ReScript](#)
 - [ClojureScript](#)
 - neueres JavaScript zu älterem JavaScript
- Transpiler (z.B. [Babel](#))

WEB-ASSEMBLY

- Neuere Entwicklung
- Virtuelle Maschine
- Diverse Sprachen zu Web-Assembly kompilierbar

ÜBERSICHT

- Worum geht's in WBE?
- Organisatorisches
- Internet und WWW
- Client-Server-Architektur
- Die Sprachen des Web: HTML, CSS, JavaScript
- **Web-Standards und APIs**

WORLD WIDE WEB CONSORTIUM

- Kurz: W3C
- Standardisiert Technologien des World Wide Web
- Gegründet 1994 am MIT
- Gründer und Vorsitzender: Tim Berners-Lee

Leider schwindet in letzter Zeit der Einfluss des W3C auf die weitere Entwicklung des Web, da die Browser-Hersteller (dominanter Player: Google) mehr und mehr die Richtung vorgeben

WHATWG

- Web Hypertext Application Technology Working Group
- Gründer: Apple, Mozilla, Opera, später: Microsoft, Google
- Grund: Unzufriedenheit mit W3C-Ausrichtung
- Eigene Standards der WHATWG
- Zeitweise parallel: HTML des W3C, HTML der WHATWG
- Heute: **HTML Living Standard** der WHATWG

<https://html.spec.whatwg.org/multipage/>

TECHNOLOGIEN

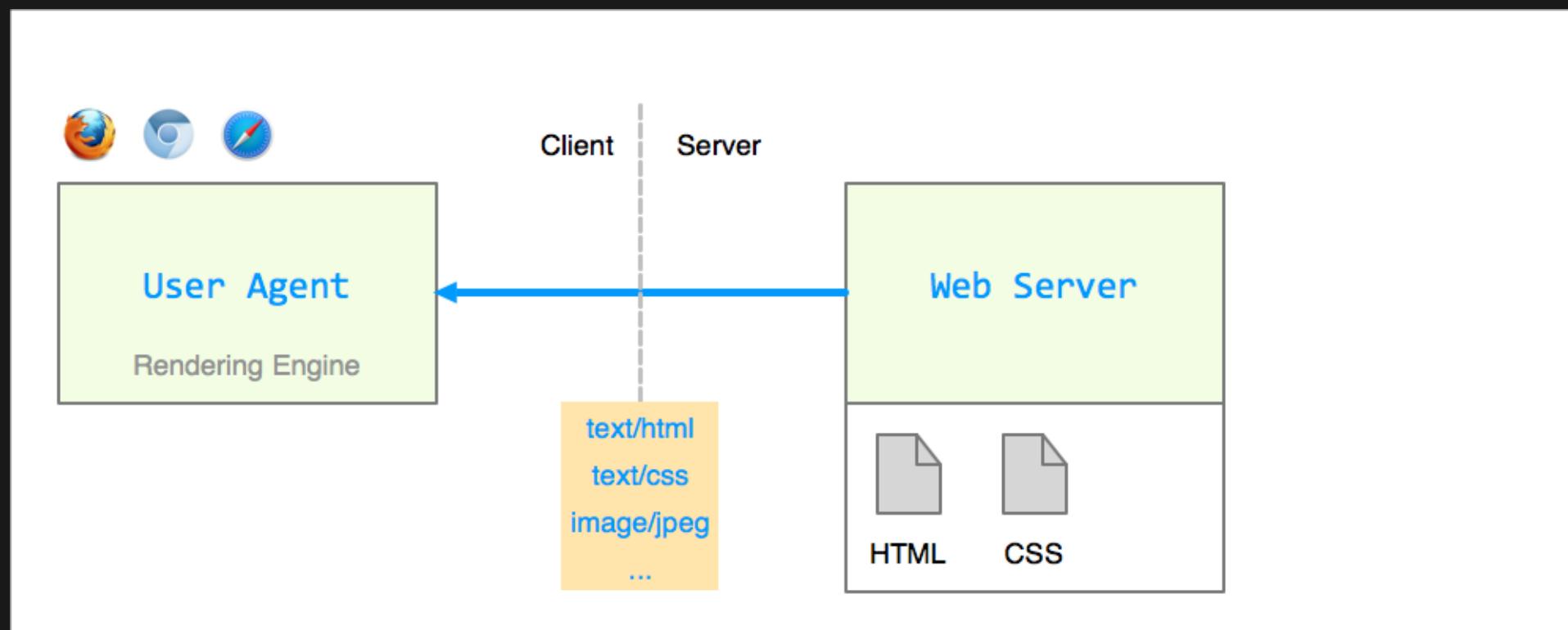
Client-seitig

- Beschränkt auf das, was der Browser kann
- HTML + CSS + JavaScript + noch ein paar Sachen
- → Front-end Entwickler

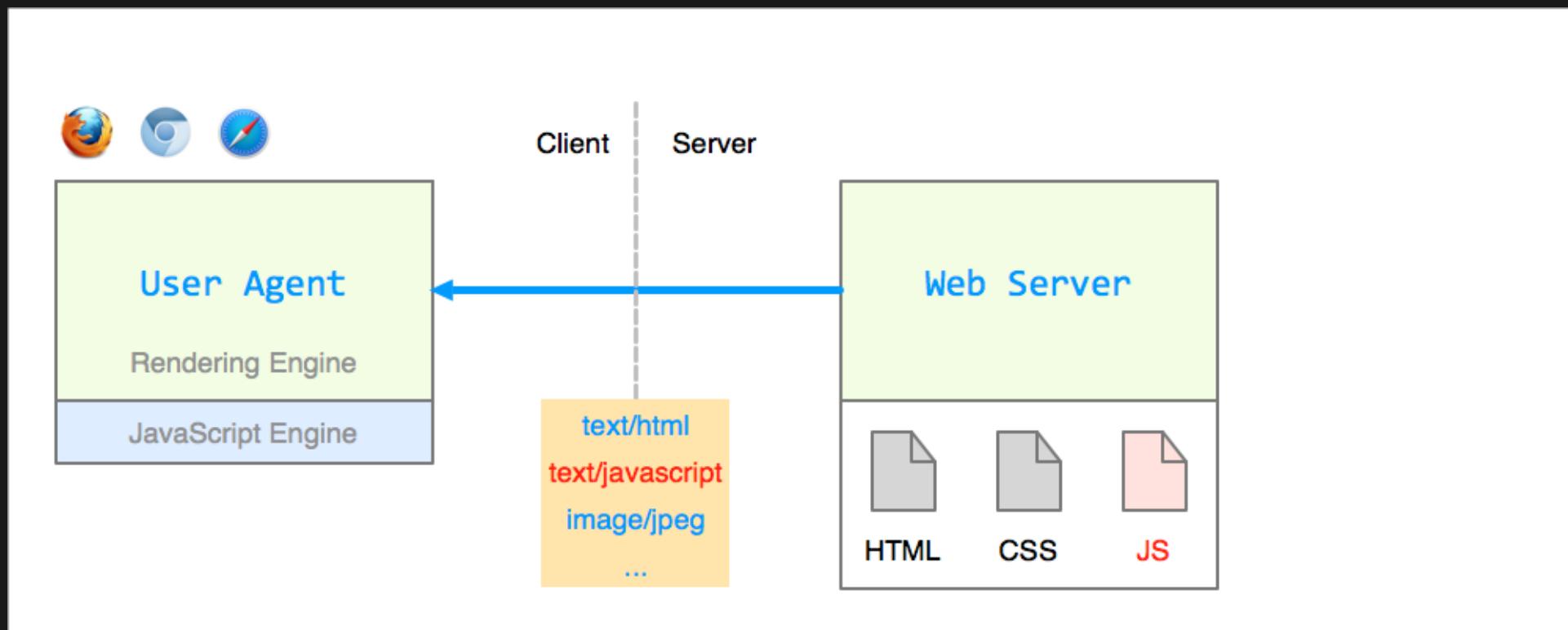
Server-seitig

- Praktisch unbeschränkt: Plattform, Programmiersprache, ...
- Erzeugt und gesendet wird das, was der Browser kann
- → Back-end Entwickler

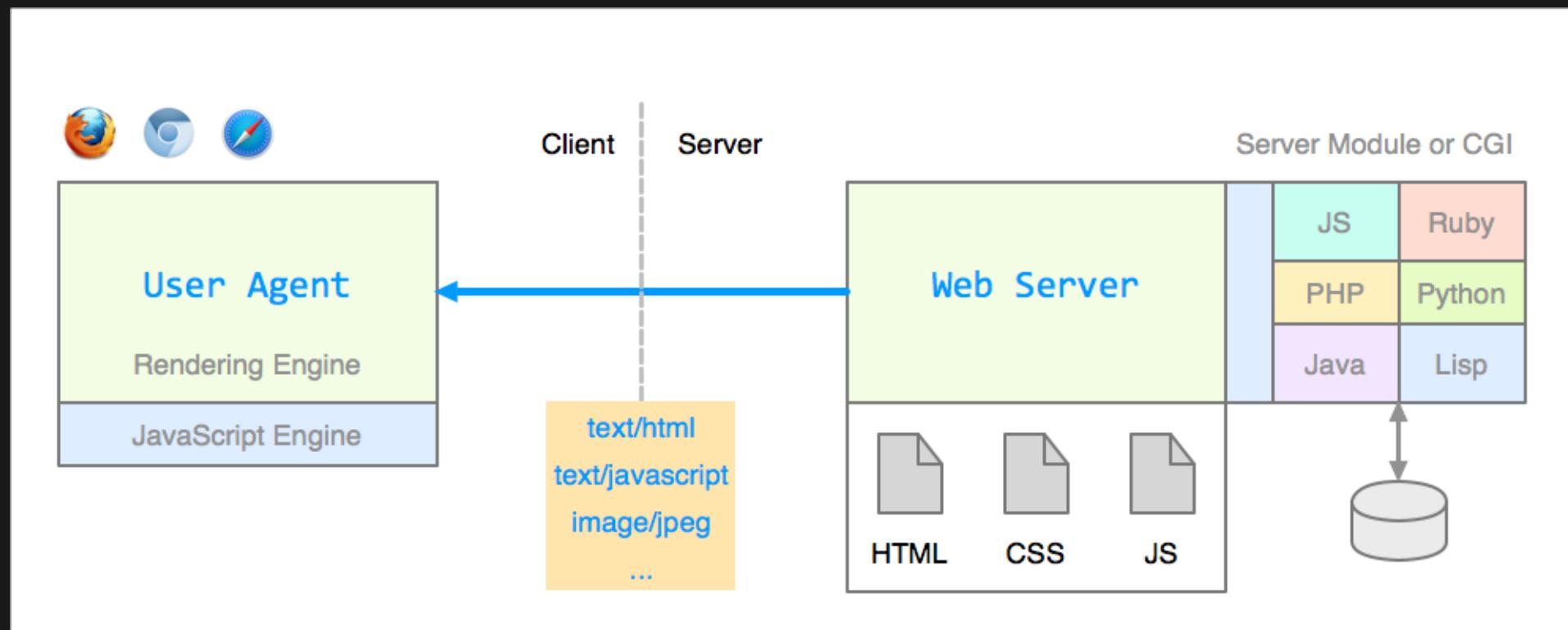
STATISCHE WEBSEITEN



CLIENTSEITIGE PROGRAMMLOGIK



SERVERSEITIGE PROGRAMMLOGIK



ENTWICKLUNG

- Statische Webseiten
- Generierte Inhalte (CGI z.B. Shell Scripts, Perl)
- Serverseitig eingebettete Scriptsprachen (PHP)
- Client Scripting und Applets (JavaScript, Java Applets, Flash)
- Enterprise Application Server (Java, Java EE)
- MVC Serveranwendungen (Rails, Django)
- JavaScript serverseitig

UMFANGREICHES GEBIET

The screenshot shows the 'Can I use' website interface. At the top, there's a navigation bar with tabs for Home, News, and a date (June 26, 2020 - New feature: Portals). Below the navigation is a large orange header with the text 'Can I use'. To the right of the header are 'Compare browsers' and 'Settings' buttons. The main content area is divided into several sections:

- CSS**: A list of CSS pseudo-elements and pseudo-classes.
- HTML5**: A list of HTML5 features and elements.
- SVG**: A list of SVG features.
- JS API**: A list of JavaScript API features.

On the right side of the page, there's a sidebar titled 'Can I use' with sections for CSS, HTML5, and SVG, each listing numerous specific features. At the bottom of the sidebar, there's a section for 'All JS API features'.

- CSS**
 - ::first-letter CSS pseudo-element selector
 - ::placeholder CSS pseudo-element
 - ::selection CSS pseudo-element
 - :dir() CSS pseudo-class
 - :has() CSS relational pseudo-class
 - :in-range and :out-of-range CSS pseudo-classes
 - :is() CSS pseudo-class
 - :placeholder-shown CSS pseudo-class
 - @font-face Web fonts
 - Blending of HTML/SVG elements
 - calc() as CSS unit value
 - Case-insensitive CSS attribute selectors
 - ch (character) unit
 - CSS 2.1 selectors
 - CSS ::marker pseudo-element
 - CSS :any-link selector
- HTML5**
 - accept attribute for file input
 - Add to home screen (A2HS)
 - async attribute for external scripts
 - Attributes for form submission
 - Audio element
 - Audio Tracks
 - Autofocus attribute
 - Canvas (basic support)
 - Canvas blend modes
 - classList (DOMTokenList)
 - Color input type
 - contenteditable attribute (basic support)
 - Custom Elements (V1)
 - Custom protocol handling
 - Datalist element
 - dataset & data-* attributes
 - Date and time input types
- SVG**
 - SVG (basic support)
 - SVG effects for HTML
 - SVG filters
 - SVG fonts
 - SVG fragment identifiers
 - SVG in HTML img element
 - SVG SMIL animation
 - All SVG features**
- JS API**
 - AbortController & AbortSignal
 - Accelerometer
 - Ambient Light Sensor
 - Auxclick
 - Base64 encoding and decoding
 - Basic console logging functions

WEB OMNIPRÄSENT

- Web-Seiten oder Web-Apps im Browser
- Auf Desktop, Notebook, oder Smartphone
- Native Mobilapps mit Web-Kern (PhoneGap, Cordova)
- Native Desktop-Apps auf Web-Basis (Electron)
- Apps, welche REST-basierte Webservices nutzen

ZIELE WBE (WH)

- JavaScript gut verstehen und einsetzen können
- Nebenbei: HTML und CSS verstehen (Selbststudium)
- Auswahl von Web-APIs kennen und einsetzen können
- Eigenes kleines Web-Framework erstellen können

ABGRENZUNG

- Schwerpunkt Front-end, wenig Back-end
- Beschränkung auf ausgewählte Standards und Web-APIs
- Nur ansatzweise: mobiles Web und zugehörige APIs
- Kein Thema: Gestaltung, Usability und Accessibility

FRONT-END DEVELOPMENT

- A practical guide to learning front end development for beginners
<https://www.freecodecamp.org/news/a-practical-guide-to-learning-front-end-development-for-beginners-da6516505e41/>
- Developer Roadmap
<https://github.com/kamranahmedse/developer-roadmap>
- Learn to become a modern Frontend Developer in 2020
<https://medium.com/@kamranahmedse/modern-frontend-developer-in-2018-4c2072fa2b9c>
- MDN: Learn Web Development
<https://developer.mozilla.org/en-US/docs/Learn>
- Frontend Masters Books
<https://frontendmasters.com/guides/>

