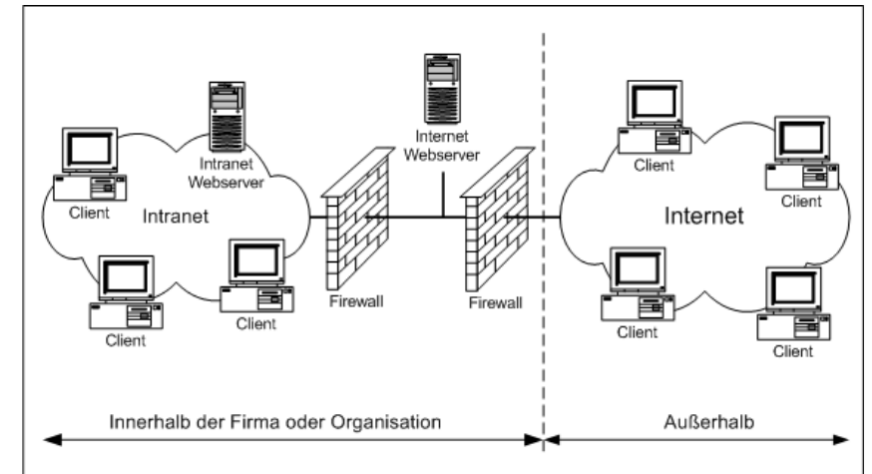




**WORKSHOP**  
**WebSecurity**  
Mathias Knoll

# Internet

- 🌐 **Dezentrales Computernetzwerk** von Rechnern
- 🌐 **Netz aus Subnetzen**
  - Netzwerkknoten vieler ISP zum effizienteren Datenaustausch
  - IXP (Internet Exchange Point), NAP (Network Access Point)
  - Firmen & Organisationen betreiben lokale Netze via Subnetting
- 🌐 **Keine zentrale Kontrolle**
- 🌐 **Dienste**
  - **World Wide Web (WWW)** | HTTP+ URLs+ Ressourcen (Hypertext, Hypermedia/Files)
  - **Email** | SMTP, POP, IMAP
  - **Dateitransfer** | FTP, SFTP, NFS, ...
  - **Fernzugriff** | SSH, telnet



# Organisationen

## **W3C** (World Wide Web Consortium)

- Seit 1994
- Definition von WWW-Standards (HTML, CSS, XML, SVG uvm.)
- Ablauf zum Standard (= W3C-Recommendation)
  - Working Draft → Last Working Draft → Candidate Recommendation → Proposed Recommendation → Recommendation
  - Finanzierung über Mitgliederorganisationen & Spenden

## **WHATWG** (Web Hypertext Application Technology Working Group)

- Browserhersteller: Apple, Mozilla, Google, Microsoft
- Etablierung aufgrund langsamer Recommendation-Prozesse des W3C
- HTML5 bzw. HTML Living Standard

# URI & URL

 **URI (Uniform Resource Identifier)** `<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"`

- RFC3986
- Eine „compact sequence of characters that identifies an abstract or physical resource“
- muss nicht physisch existieren, aber eindeutig sein
- sieht oftmals aus wie URL

 **URL (Uniform Resource Locator)** 

- Identifikation von Online-Inhalten
- Ist eine Spezialform eines URI
- Aufbau:  
`<Protocol>:// [<Username>:<Password>@] <Host> [:<Port>] [/<Path> [ ?<Query> [#Fragment] ] ]`

# URL Encoding

- 🌐 ASCII-Basiszeichensatz
- 🌐 Reservierte Zeichen
- 🌐 Lösung:
  - Percent-Encoding z.B. Leerzeichen à%20 (vgl. RFC3986)
  - Alternativ internationale URLs (vgl. RFC3987)
  - Tipp: Vermeiden Sie Sonderzeichen in URLs und bilden Sie Leerzeichen als "-" ab. (Nicht "\_")

# Hypertext Transfer Protocol (HTTP)

- 🌐 Übertragung von Inhalten über das Netzwerk
- 🌐 Beispiel: Ein Client fordert über das Internet von einem Webserver eine Website an
- 🌐 Zustandslos
  - Requests stehen für sich alleine
  - Sitzungsidifikation mittels Session-IDs über z.B. Cookies
- 🌐 Aufbau:
  - Header: Metainformationen
  - Body: Payload (HTML,...)

**Site** | Die gesamte Webpräsenz unter einer Domain  
**Webseite** | Eine einzelne Seite unter einer URL  
**Homepage** | Die Startseite einer Website

# Rich Internet Applications

Desktop  
Anwendungen

- Frameworks
- Bibliotheken
- Sensoren

Internetdienste

Single Page  
Anwendung (SPA)  
Progressive Web  
App (PWA)

Mobile  
Anwendungen

Kommunikations-  
technologien

Web  
Anwendungen

- Lokale Logik
- Lokale Persistenz
- Verschiedene Geräte
- Verschiedene Plattformen



## Werkzeuge

- 🌐 Webstorm oder
  - <https://www.jetbrains.com/webstorm/>
- 🌐 Visual Studio Code
  - <https://code.visualstudio.com/download>
- 🌐 NodeJS
  - <https://nodejs.org/>
  - Version anzeigen  
`node -v`
  - Installierte Pakete  
`npm list -g -depth=0`
  - Installieren/Aktualisieren von Paketen  
`npm install -g <package>`



Visual Studio Code

```
$ node -v  
v16.3.0
```

```
$ npm list -g -depth=0  
npm notice  
npm notice New minor version of npm available! 7.15.1 -> 7.20.5  
npm notice Changelog: https://github.com/npm/cli/releases/tag/v7.20.5  
npm notice Run npm install -g npm@7.20.5 to update!  
npm notice  
/development/npm  
+-- cordova@10.0.0  
+-- ionic@5.4.16  
+-- native-run@1.3.0  
`-- npm@6.14.11
```



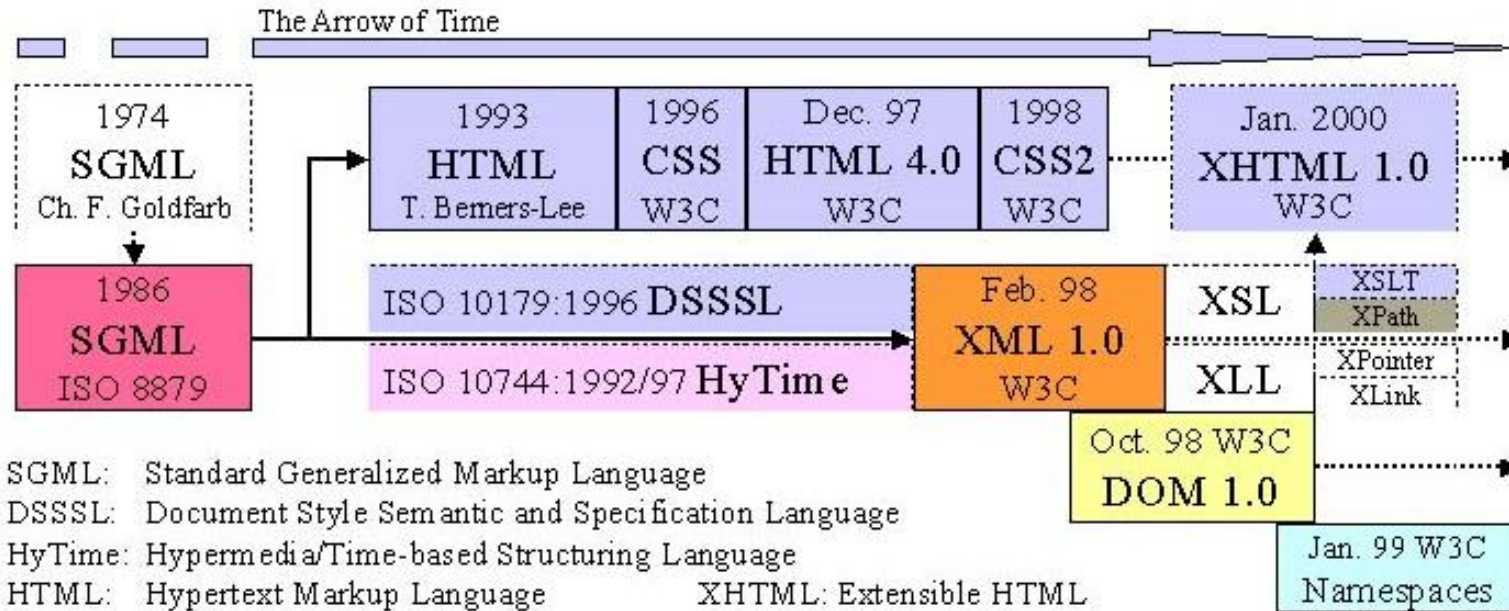




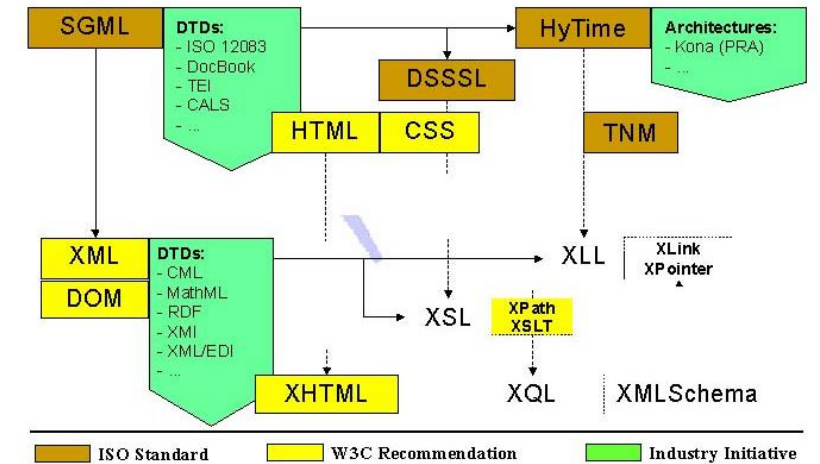
# Hypertext Markup Language (HTML)

## Cascading Stylesheets (CSS)

# Standards



- SGML: Standard Generalized Markup Language
- DSSSL: Document Style Semantic and Specification Language
- HyTime: Hypermedia/Time-based Structuring Language
- HTML: Hypertext Markup Language
- CSS: Cascading Style Sheets
- XML: Extensible Markup Language
- XLL: XML Linking Language
- ISO: Int. Organization for Standardization
- XHTML: Extensible HTML
- DOM: Document Object Model
- XSL: XML Stylesheet Language
- XQL: XML Query Language
- W3C: World Wide Web Consortium



Quelle: <https://www.jmir.org/2000/2/e12/>

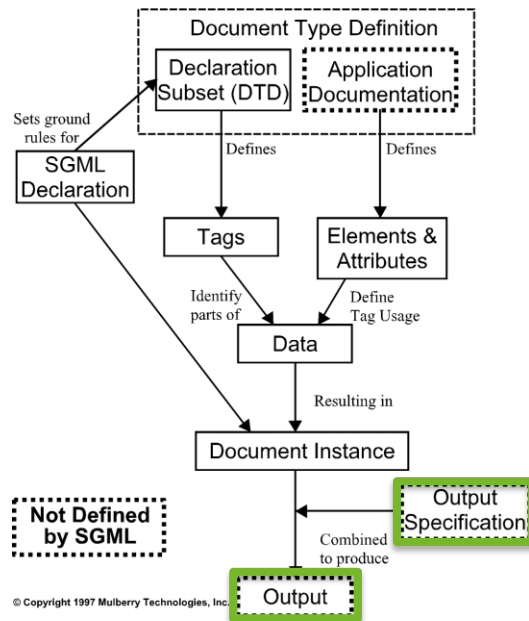
# Bekannte Sprachen im Vergleich

Beschreibung, wie eine Sprache definiert wird

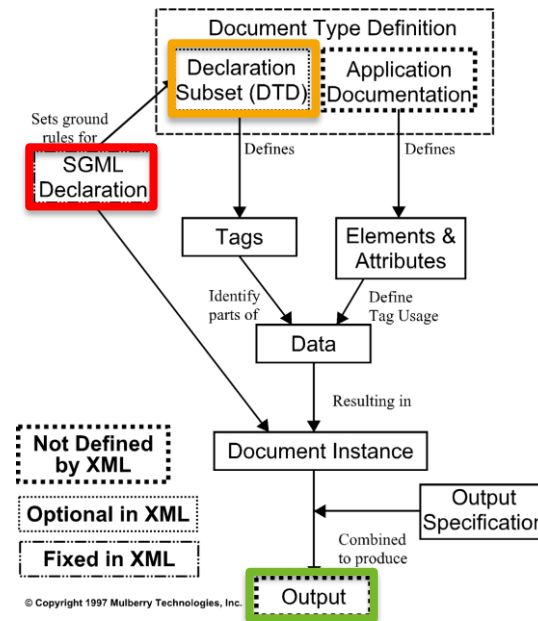
Einfacher gemachte Teilmenge von SGML

Angewandtes SGML

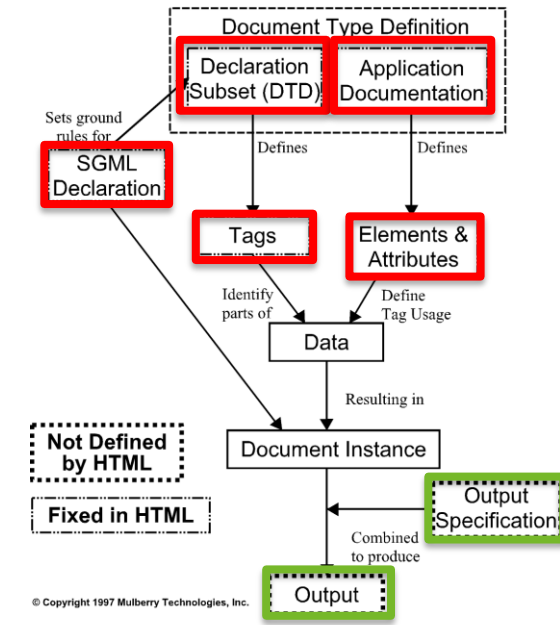
## SGML Document Components



## XML Document Components



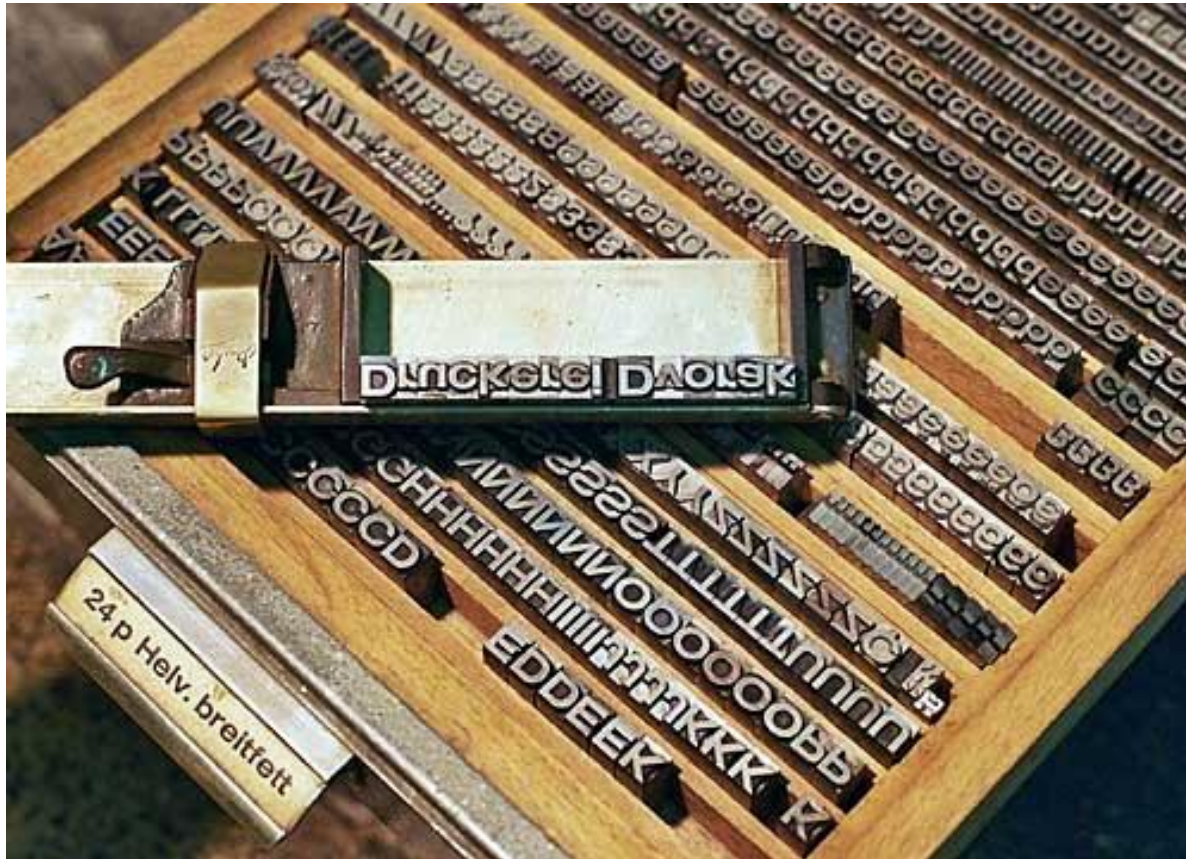
## HTML Document Components



# Hypertext

- 🌐 **Keine sequentielle Präsentationsreihenfolge**
- 🌐 **Dokument** als Menge **verlinkter Ressourcen**
  - Format der Ressourcen
  - Format der Links
- 🌐 **Erste Systeme** waren rein **textbasiert**
- 🌐 **Zunehmende Bedeutung des Betrachters**
  - keine eindeutige Reihenfolge vorgegeben
  - aktive **Navigation** statt passiver **Präsentation**
- 🌐 Wenn auch **Multimedia-Dokumente** (nicht rein textbasiert: Film, Animation, Audio) eingesetzt werden, spricht man auch von **Hypermedia**

# Hypertext Markup Language (HTML)



```
<HEADER>
<TITLE>The World Wide Web project</TITLE>
<NEXTID N="55">
</HEADER>
<BODY>
<H1>World Wide Web</H1>The WorldWideWeb (W3) is a wide-area<A
NAME=0 HREF="WhatIs.html">
hypermedia</A> information retrieval
initiative aiming to give universal
access to a large universe of documents.<P>
Everything there is online about
W3 is linked directly or indirectly
```

## HTML 1.0

Beschreibt den Inhalt von Webseiten. HTML ist das Format, in dem die Text- und Hypertext-Informationen im WWW gespeichert und übertragen werden und ist eine Anwendung von SGML (Standard Generalized Mark up Language)

## Versionen

HTML 1.0

HTML 4.0 Externe Stilangaben über eigene Sprache möglich (CSS)

XHTML 1.0/1.1 und **HTML 5.0**

<https://www.w3schools.com/html/default.asp>

# Hypertext Markup Language - HTML

- 🌐 Hypertext-Dokumentformat, Auszeichnungssprache
- 🌐 Wird in Dateien gespeichert oder mittels Programmen dynamisch generiert
- 🌐 Enthält <tags>, welche einerseits Strukturanweisungen, Formatanweisungen oder Verweise auf andere Ressourcen (Links) enthalten.
- 🌐 Ressourcen können Dokumente jeglicher Art sein:
  - Andere HTML Dokumente
  - Grafiken und Bilder (gif, jpeg, png)
  - Video und Sounddateien
- 🌐 Über Links können auch andere Programme aufgerufen werden (Telnet, FTP, Mail, ...).
- 🌐 Kann auch Formulare enthalten, welche Benutzereingaben erwarten und diese zur Bearbeitung an den Server schicken.
- 🌐 **WICHTIG:** Trennung von Inhalt und Darstellung

# HTML5



## Entwicklung

- Web Hypertext Application Technology Working Group (WHATWG)
- Referenz: <https://www.w3.org/TR/html50/>



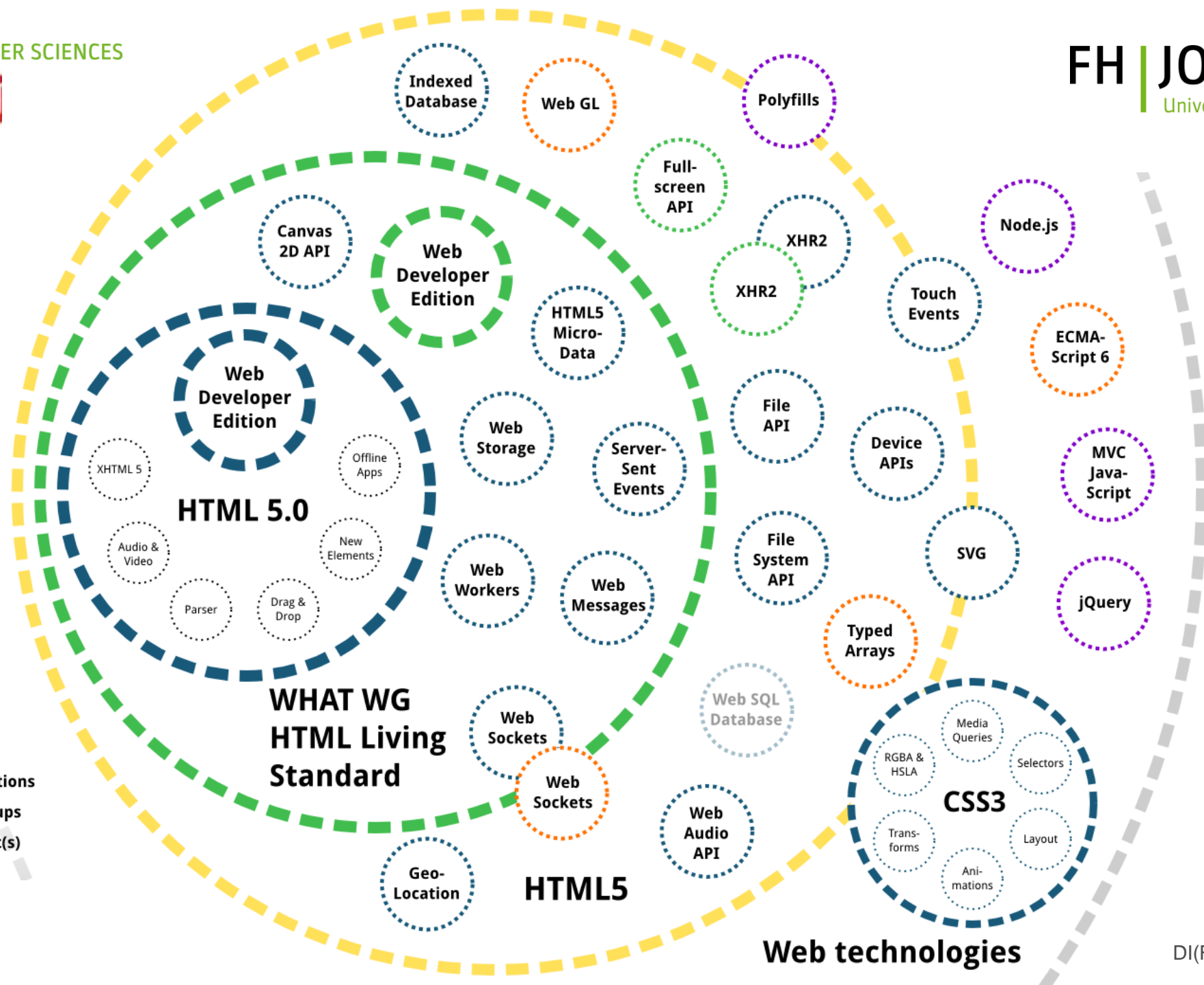
## Neue Features

- Strukturell (Neue Bereiche)
- Formulare (Elemente)
- Audio, Video & Grafik (2d/3D)



## URLs

- [https://en.wikipedia.org/wiki/Comparison\\_of\\_browser\\_engines\\_\(HTML\\_support\)#HTML5](https://en.wikipedia.org/wiki/Comparison_of_browser_engines_(HTML_support)#HTML5)
- <http://html5test.com/>
- <https://caniuse.com/>
- <https://www.w3.org/TR/2011/WD-html5-20110525/content-models.html#content-models>



- W3C Specifications
- WHATWG Specifications
- Other working groups
- Community project(s)
- Common use



# HTML | Grundsätzlicher Aufbau

```
<html>
```

```
  <head>
```

```
    Weitere Tags
```

```
  </head>
```

```
  <body>
```

```
    Weitere Tags oder Text!
```

```
  </body>
```

```
</html>
```

- Verschachtelung von Tags bildet eine Baumstruktur: **DOM (Document Object Model)**
- Tags müssen / sollten immer einen **Abschluss**tag haben!
- Wenn ein **Tag keinen Inhalt** hat, kann man den Abschluss gleich im Anfangstag einbauen:  
`<br />`

[https://www.w3schools.com/html/html\\_basic.asp](https://www.w3schools.com/html/html_basic.asp)

# HTML | Anatomie einer Seite

```

<!DOCTYPE html>

<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Titel im Fenster oder Tab</title>
  </head>
  <body>
    <header data-meinAttribut="xyz">
      Am Bildschirm gerenderte Elemente
    </header>
  </body>
</html>
  
```

**Typ**

**Struktur**

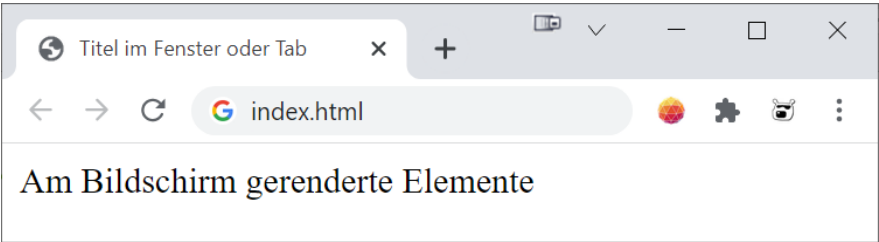
**Dokumenttypdefinitionen (DTD):**

- HTML5 hat keine DTD mehr- weil dynamischer z.B. selbstbestimmte Elemente wie eigene Attribute: `data-mathias="meine Eigenschaft"`
- Alle Vorgänger hatten ein genaues Set an Regeln.  
z.B.: `<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">`

"Kopf" mit Metainformationen zur Seite

"Körper" mit Inhalten

Kein SGML mehr .... Weil nicht per DTD abbildbar



Workshop WebSecurity

# HTML | Der Tag / Das Element

- 🌐 **Tags sind Element der Seite.**
- 🌐 Tags **beschreiben** ein Element, werden selbst **aber nicht angezeigt!**
- 🌐 Tags können weitere **beschreibende Attribute** besitzen:  
``
- 🌐 Zeichen, die zum Beschreiben der Seite verwendet werden, müssen selbst beschrieben werden. Man spricht hier von „**Entitäten**“:  
„<“ wird mit „**&lt;**“, „>“ wird mit „**&gt;**“ „maskiert.“

# HTML Elemente | Verweise (Links)

## Interne Verweise

```
<a href="seite.html" target="_self"> Zur Seite </a>
```

## Externe Verweise

```
<a href="http://extern.at/seite.html"> Zur externen Seite </a>
```

## Anker auf einer Seite

```
[http://external.at/beispiel.html]
```

```
<p>  
Das Beispiel ist mit einem Anker belegt. Dieser Text ist in der Seite  
beispiel.html unter http://external.at erreichbar!
```

```
</p>
```

```
[index.html]
```

```
<p>  
  <a href="http://external.at/beispiel.html#beispiel">Zum Beispiel</a>  
</p>
```

„target“ öffnet den Verweis mit ...  
... „\_blank“ auf einer neuen Seite  
... „\_self“ in der aktuellen Seite

# HTML Elemente | **Formulare**

🌐 **Benutzereingabe** aufnehmen und versenden.

🌐 **Tag:** <form> ...Elemente... </form>

🌐 **Attribute** des Formulars:

- **action**="<URL oder relative Angabe zu einer Seite>" - verarbeitende Seite, an welche die Eingaben übergeben werden
- **method**="<get | post>"
  - **get** → Formulardaten als Parameter nach dem Verweis von „action“
  - **post** → spezielles versenden – bei vielen Formulardaten zu verwenden!
- **accept-charset**="UTF-8" - Zeichenkodierung der Daten

🌐 **Elemente**

- <input>, <textarea>, <select>, <button>

[https://www.w3schools.com/html/html\\_forms.asp](https://www.w3schools.com/html/html_forms.asp)

# HTML Elemente | Formularbeispiel

Formular

Texteingabefeld

Auswahl

Textfeld

Buttons

```
<form action="http://mysite.at/cgi-bin/test.pl">
  <table border="0" cellpadding="5" cellspacing="0" bgcolor="#E0E0E0">
    <tr>
      <td align="right">Name:</td>
      <td><input name="Name" type="text" size="30" maxlength="30"></td>
    </tr>
    <tr>
      <td align="right">Ort:</td>
      <td>
        <select name="Ort" size="3">
          <option>Wien</option>
          <option>Paris</option>
        </select>
      </td>
    </tr>
    <tr>
      <td align="right" valign="top">Kommentar:</td>
      <td><textarea name="Text" rows="10" cols="50"></textarea></td>
    </tr>
    <tr>
      <td align="right">Formular:</td>
      <td>
        <input type="submit" value=" Absenden " >
        <input type="reset" value=" Abbrechen " >
      </td>
    </tr>
  </table>
</form>
```

# HTML Elemente | Zusammenfassung

## Der „Tag“

- `<tag> Irgendetwas </tag>`
- Irgendetwas ist Text oder wieder ein weiterer „Tag“.
- „Tags“ können Attribute haben. Beispiel: `<tag width="100"></tag>`
- „Tags“ beschreiben ein Element werden aber selbst nicht angezeigt.
- Beispiel: `<font color="red"><b>fett</b> nicht fett</font>`

[https://www.w3schools.com/html/html\\_entities.asp](https://www.w3schools.com/html/html_entities.asp)

## Die „Entität“ - „maskierte“ Zeichen

- &Kürzel des Zeichens;
- Beispiele: `&amp;` ... `&`   `&gt;` ... `>`   `&lt;` ... `<`   `&nbsp;` ... `„ „`

[https://www.w3schools.com/html/html\\_entities.asp](https://www.w3schools.com/html/html_entities.asp)

# Cascading Style Sheets (CSS)



- 🌐 Entwicklung
  - World Wide Web Consortium
  - Referenz: <https://www.w3.org/TR/CSS/#css>

- 🌐 Einige Features
  - Media Queries
  - Layouts (Grid-, Box-, Flex- usw.)
  - Animationen, Webfonts, usw.



- 🌐 URLs
  - <https://enjoycss.com>
  - <http://css3test.com/>
  - <http://www.csszengarden.com/>
  - [https://en.wikipedia.org/wiki/Comparison\\_of\\_browser\\_engines\\_\(CSS\\_support\)](https://en.wikipedia.org/wiki/Comparison_of_browser_engines_(CSS_support))

*Sass*  
Syntactically **Awesome** Style Sheets  
CSS pre-processor  
Scriptsprache, welche  
SassScript in CSS  
transpiliert.  
<https://sass-lang.com/>



# CSS | Formatierung

- Formatangaben für (X)HTML Elemente
  - Farben, Größen, Positionen, Bilder, Abstände, Sichtbarkeit, ...

- Einbindung in HTML
  - Innerhalb des Seitenkopfes (head)
 

```
<style type="text/css"> ... </style>
```
  - In einer oder mehreren externen Dateien
 

```
<link rel="stylesheet" type="text/css" href="formate.css" />
```
  - Innerhalb eines Tags (style-Attribut)
 

```
<p style="....."> ... </p>
```

- Syntax:
 

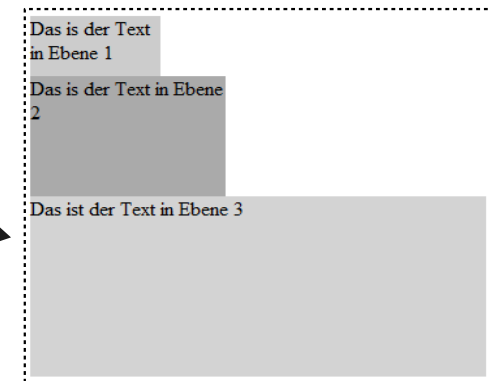
```
[Selektor] {[Eigenschaft] : [Wert];}
```

  - Selektor:
    - Bezug zu dem gewünschten Element
    - Zum Beispiel der Tagname p
  - Eigenschaft:
    - Formatangabe, welche gesetzt werden soll
    - Zum Beispiel eine Farbe: #DDEE00

```
css/layout.css
/* In Ebene 1 ist eine Klasse definiert */
div.ebene1 {
    width: 100px; height: 50px;
    background-color: #CCCCCC;
}
/* In Ebene 2 ist eine Id definiert */
#ebene2 {
    width: 150px; height: 100px;
    background-color: #AAAAAA;
}
```

```
beispiel.html
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<meta http-equiv="Content-type" content="text/html; charset=UTF-8" />
<title>CompanyName - PageName</title>
<link rel="stylesheet" type="text/css" href="css/layout.css" />
</head>
<body>
<div class="ebene1">
    Das is der Text in Ebene 1
</div>
<div id="ebene2">
    Das is der Text in Ebene 2
</div>
<div style="width:350px;height:150px;background-color:lightgrey;">
    Das ist der Text in Ebene 3
</div>
</body>
</html>
```

Im Browser:



# CSS | Struktur

## Prioritäten

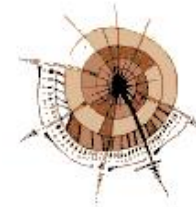
- Nach Element Id: #[Element-Id]
  - Element: `<p id="spezial">Test</p>`
  - Styleangabe: `#spezial { color:red; }`
- Nach Klassen: [Tagname].[Klassenname]
  - Element: `<p class="spezial">Test</p>`
  - Styleangabe: `p.spezial { color:red; }`
- Pseudoklasse: [Tagname]:[Pseudoklasse]
  - Für Elementeigenschaften, die nicht eindeutig durch ein HTML Element beschreiben lassen: Typisch für Links: `a:link`, `a:visited`, `a:hover`

## Verschachtelung

- Angabe eines Stile für mehrere Elemente: Selektoren mit Beistrich getrennt
- Direktes Kindelement ansprechen: Selektoren mit Leerzeichen getrennt:  
Beispiel: `td p { color:red; }`
- Irgendein Kindelement darunter ansprechen: Selektoren mit \* getrennt  
Beispiel: `table * p { color:red; }`

## Wichtige Webseiten

-  <http://www.w3schools.com/>
  - Anleitungen zum Erstellen von Webseiten
-  <http://de.selfhtml.org/>
  - DAS Kompendium über
  - Auszeichnungssprachen im Web
-  <http://www.csszengarden.com/>
  - Was alles mit Stilangaben machbar ist
-  <http://www.w3.org/>
  - Gremium zur Standardisierung der das
  - World Wide Web (WWW) betreffenden Techniken
  - <http://validator.w3.org/>
  - <http://jigsaw.w3.org/css-validator/>





# JavaScript & TypeScript

# Anwendungsgebiete

- 🌐 **Webseiten benutzerfreundlich** und **funktionell** gestalten (wie Desktop-Applikationen)
- 🌐 **Plausibilitätsprüfung** (Validierung) von Formular-eingaben vor dem Absenden
- 🌐 **Browsererkennung** (Browserweiche)
- 🌐 **Multimedia** (Slide Shows, Games, etc.)
- 🌐 Dynamische Manipulation über **DOM**
- 🌐 Senden und Empfangen von Daten, ohne dass der Browser die Seite neu laden muss mit **AJAX / Fetch**

# Was ist JavaScript?

- 🌐 **Soll wie Java aussehen**
- 🌐 **Skriptsprache vornehmlich für Clients (Browser)**
- 🌐 **ECMA Standard (seit 1997)**  
heute ECMAScript 2018  
<https://kangax.github.io/compat-table/es2016plus>
- 🌐 **Sandboxumgebung \***
  - Zugriff auf Browser (-Objekte) beschränkt
  - Kein Zugriff auf Dateisystem
  - Kein Lesen oder Schreiben von Dateien(\* Umgehung über Sicherheitsbeschränkungen im Browser u.U. möglich)

- 🌐 **Syntax ist Java und C ähnlich**
- 🌐 **Programmiersprache ist**  
Prozedural – Objektorientiert -  
Funktional
- 🌐 **Dynamische Typisierung**
- 🌐 **Einsatzgebiet: Client- u.**  
Serverseitig

*serverseitiges JavaScript: Node.js*

*ECMA  
European Computer Manufacturers Association*

# Was ist TypeScript

- Von Anders Hejlsberg, Microsoft, 2021 entwickelt
- Sprache für **typisiertes JavaScript**  
auch "überlagertes JavaScript" genannt
- Wird **in Standard-JavaScript "transpiliert"** und braucht dazu ein eigenes Programm, einen "**Transpiler**"
- Üblicherweise haben **TS Dateien \*.ts als Erweiterung** und es werden \*.js Dateien daraus erzeugt
- **JavaScript Bibliotheken und –Frameworks** können durch Deklarationsdateien **ermöglicht** werden
- **Neuere JavaScript Funktionen** werden unterstützt (z.B.: async)

*"TypeScript is JavaScript  
with syntax for types."  
-- <https://www.typescriptlang.org/>*

**JavaScript** *ideal* für kleine Projekte

- Schwach typisiert
- Mehrere Paradigmen
- siehe Websprachen!

**TypeScript** *ideal* für große Projekte

- Superset von JavaScript - transpiliert zu JavaScript
- Typen ("TypeSafe") & Interfaces
- Prototype Pattern

# Javascript | Einbettung in HTML: Scriptbereich

```
<html>
  <head>
    <title>Eingebaut</title>
    <script type="text/javascript">
      <!--
      alert („JavaScript at work!“);
      -->
    </script>
  </head>
  <body>
    Javascript Beispiel
  </body>
</html>
```

*Ältere Browser, die JS nicht kennen ignorieren Code unter Kommentaren und interpretieren ihn nicht irrtümlich als Text innerhalb der HTML-Datei! – kaum mehr der Fall...*



# Javascript | Einbettung in HTML: Datei

```
<html>
  <head>
    <title>Separate Datei</title>
    <script src="alert.js" type="text/javascript"></script>
  </head>
  <body></body>
</html>
```

Inhalt von alert.js:

```
// Kommentar
alert („Externe JavaScript-Datei at Work!");
```

# Javascript | Einbettung in HTML: Events

```
<html>
  <head>
    <title>Als Event</title>
  </head>
  <body>
    
    
  </body>
</html>
```

*Events*

# Javascript | Event Handler

## Maus-Ereignisse

- **onmouseover**
- **onmouseout**
- **onmousemove** Mauszeiger bewegen
- **onmousedown**
- **onmouseup**
- **onclick**
- **ondblclick**

## Tastatur-Ereignisse

- **onkeydown**
- **onkeyup**
- **onkeypress**



## Formularereignisse

- **onsubmit**
- **onreset**
- **onselect**  
Inhalt eines Formularelements mit der Maus markieren
- **onchange**  
Ändern des Inhaltes eines select-Feldes

## Weitere Ereignisse

- **onload**  
HTML-Seite wurde komplett(!) geladen
- **onunload**  
HTML-Seite wurde verlassen
- **onabort**  
Das Laden der HTML-Seite wurde abgebrochen

## Allgemeine Ereignisse

- **onfocus**  
Wenn ein Element aktiviert wird
- **onscroll**  
Wenn ein Element mit der Maus gescrollt werden kann
- **onblur**  
Wenn ein zuvor aktiviertes Element verlassen wird

*Es gibt sicher noch ein paar mehr ...*

# Javascript | Überblick Variablen

🌐 Zuweisung

```
a = 2; // Global ("window" Objekt Attribut)
var b = "x"; // Lokal innerhalb einer Funktion
let c = "y"; // Lokal innerhalb von Blöcken
// Global, wenn außerhalb Funktion
```

🌐 Schwach, dynamisch typisiert

```
typeof(a); // number
typeof(b); // string
```

und *boolean, function, object, undefined, null, symbol*

🌐 **Number**

In der Regel Double Precision Float  
bei Bitoperationen 32Bit Integer

🌐 **String**

Unicode/UTF16 – einmal erstellt nicht veränderbar

*Achtung!  
Globale Variablen können  
auch in Funktionen  
deklariert werden!*

# TypeScript | Überblick Typen

- `boolean`
- `number`
- `string`
- `number[]` oder `Array <number>`
- Tupel: Arrays mit unterschiedlichen Typen: `[string, number]`
- Mehrere Konstanten (enums):  
`enum Color {red=1, green, blue}`  
`let color:Color = Color.green; // 2`
- `any` (unbestimmter Typ); `any[]`
- `void` (leer) ist `null` oder `undefined`  
Letztere sind auch definiert
- `never` (bei Funktionen: nie ein return)
- `object` (kann auch null sein)

```
function helloworld(degree_programme:string) {  
    return "Hello" + degree_programme;  
}
```

```
let dp = {};
```

```
document.body.innerHTML = helloworld(dp);
```

! Error:(7, 38) TS2345: Argument of type '{}' is not assignable to parameter of type 'string'.

Überprüfung (Assertion):

```
let zeichen:number = (<string> wert).length;  
let zeichen:number = (wert as string).length;  
(Keine Auswirkungen beim transpilieren.)
```

# TypeScript | Eigenen Datentypen deklarieren

## 📦 Eigene Typen vs. 📦 Standardtypen

- Einfach oder kombiniert mit '|' (oder)

```
type meinSpezifischerTyp = number; type meinSpezifischerTyp = string | number;
type meinUnbestimmterTyp = any;
```

- Generisch

```
type meinGenerischerTyp<Type> = Type;
```

- Objekte (Schlüsselwort 'type' oder 'interface' kann verwendet werden!)

```
type Pizza = {
  name: string;
  typ: string;
  durchmesser: number
};
interface Pizza2 extends Pizza = {
  notiz?: string;
  readonly rating: number
};
```

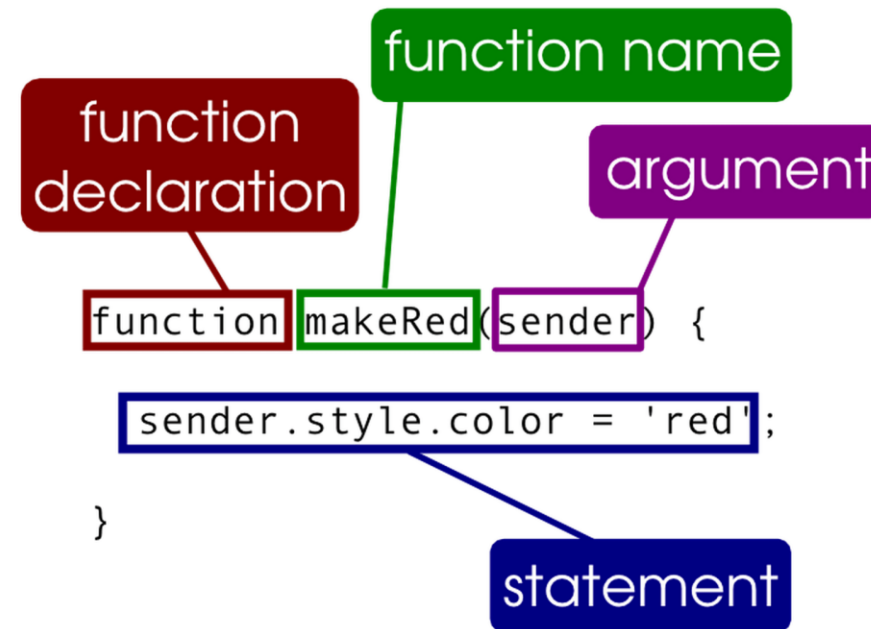
*Man kann Typen auch erweitern ('extends')*

*Optionale Eigenschaften können mit '?' markiert werden!*

*Eigenschaften, die nicht mehr geändert werden können mit 'readonly' markiert werden (in der IDE beim Kompilieren relevant)*

# Javascript | Funktionen in Javascript

- 🌐 Funktionen werden **wie Objekte** behandelt  
(*First-Class-Functions*) und merken sich ihren **Erstellungskontext** (*Closure*)
- 🌐 Funktionen sind **Eigenschaften von Objekten!**
- 🌐 Funktionen können Funktionen **enthalten**
- 🌐 Funktionen können **als Parameter Funktionen** erhalten



[https://www.w3schools.com/js/js\\_functions.asp](https://www.w3schools.com/js/js_functions.asp)

# Javascript | Funktionen (*einfach*)

## Funktionen selbst definieren in HTML

```
<script src="alert.js" type="text/javascript">  
    function simple(zahl) { ... }  
</script>
```

```
<input type="button" value="Simple" onclick="simple(22307)"/>
```

## Funktionen mit Rückgabewert

```
function simple(zahl) {  
    var ergebnis = zahl; // Lokale Variable  
    return ergebnis;  
}
```

```
var resultat = simple(22307); // Globale Var.
```



# Javascript | Funktionen (*komplex*)

```
function makeSandwich () {  
    var ingredient = 'bread';  
    function make (filling) {  
        return ingredient + ' and ' + filling;  
    }  
    return make ('ham');  
}
```

*Eine Funktion in der Funktion ...*

*... und diese wird zurückgegeben ;-)*

```
makeSandwich (); // Gibt „bread and ham“ zurück
```

Adaption des unteren Teils des Snippets möglich:

```
    return make;  
}
```

*Die Funktion wird in eine variable gespeichert ...*

```
var f = makeSandwich ();  
f ('ham'); // Gibt „bread and ham“ zurück
```

# Javascript | Objekte (object)

- 🌐 Container für **Schlüssel-Wert-Paare**
- 🌐 Der **Schlüssel** bezeichnet die **Eigenschaft** oder eine **Methode** des Objekts
- 🌐 Der **Wert** kann ein **Literal**, eine **Funktion** oder ein anderes **Objekt** sein
- 🌐 Objekte können auf verschiedene Weisen erstellt werden
  - **Konstruktorfunktion**
  - **Objekt-Literal-Schreibweise**
  - Methode "**create**" des Objekts "**Object**"
- 🌐 Der Zugriff auf Eigenschaften des Objekts erfolgt mittels "**."** **Notation** (wie in Java) oder die "**[]**" **Notation**

[https://www.w3schools.com/js/js\\_objects.asp](https://www.w3schools.com/js/js_objects.asp)

# Javascript | Objekte erstellen & darauf zugreifen

- 🌐 Fest implementiert
  - **String, Number, Array** oder **Math**
- 🌐 Host Objekte (Seitenstruktur)
  - **window, document** oder **form**
- 🌐 Benutzerdefinierte Objekte

```
myobject = new Object();  
oder  
myobject = {};
```

*Letztere Schreibweise wird auch bei JSON verwendet - JavaScript Object Notation*

```
pizza = {  
    name : "Diabolo",  
    preis : 7.5,  
    zutaten : ["Käse", "Tomaten", "Salami", "Pfefferoni"]  
};
```

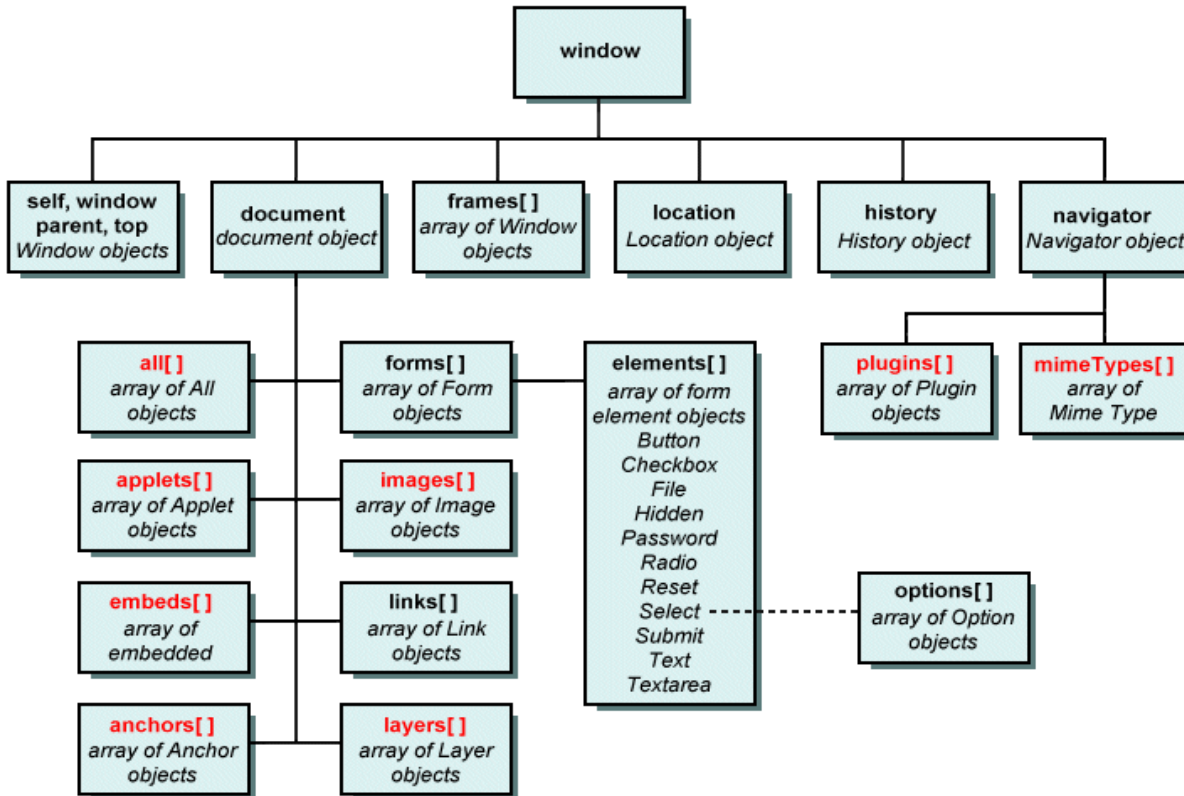
```
pizza = new Object();  
pizza.name = "Diabolo";  
pizza.preis = 7.5;  
pizza.zutaten = ["Käse", "Tomaten", "Salami", "Pfefferoni"];
```

```
pizza = new Object();  
pizza["name"] = "Diabolo";  
pizza["preis"] = 7.5;  
pizza["zutaten"] = ["Käse", "Tomaten", "Salami", "Pfefferoni"];
```

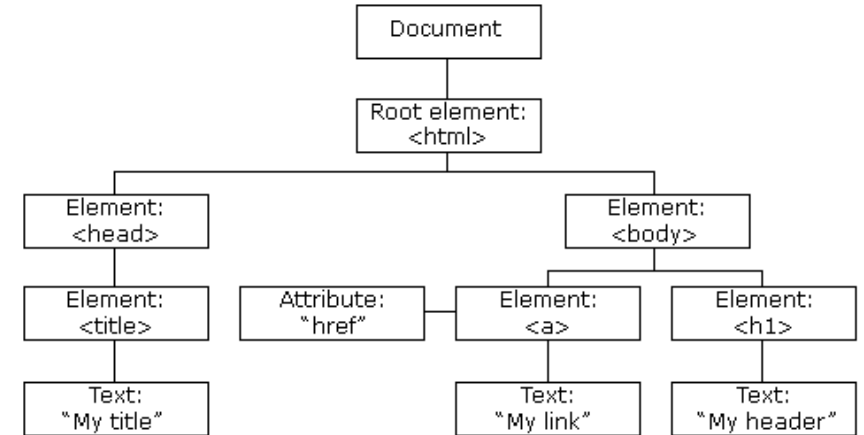
```
alert(pizza["name"]); // alert(pizza.name); // "Diabolo"  
alert(pizza["preis"]); // alert(pizza.preis); // 7.5  
alert(pizza["zutaten"][0]); // alert(pizza.zutaten[0]); // "Käse"  
alert(pizza["zutaten"][1]); // alert(pizza.zutaten[1]); // "Tomaten"
```

[https://www.w3schools.com/js/js\\_object\\_properties.asp](https://www.w3schools.com/js/js_object_properties.asp)

# Javascript | Überblick der Host Objekte



*Achtung!  
u.u. browserabhängig, was  
hier verfügbar ist!*



[https://www.w3schools.com/js/js\\_htmlDOM.asp](https://www.w3schools.com/js/js_htmlDOM.asp)

# Javascript | Arrays (**array**)

🌐 Sind **Objekte** und werden über "**new Array()**" oder die **Literal-Kurzschreibweise** instanziiert

🌐 Wichtige Funktionen:

- **concat ()** ... Elemente an bestehende Arrays anhängen
- **forEach ()** ... Wendet eine übergebene Funktion an jedes Element im Array an
- **pop ()** ... entfernt das letzte Element
- **push ()** ... fügt ein neues Element hinzu
- **reverse ()** ... kehrt die Reihenfolge um
- **sort ()** ... sortiert auf Basis einer übergebenen Vergleichsfunktion

[https://www.w3schools.com/js/js\\_arrays.asp](https://www.w3schools.com/js/js_arrays.asp)

# Javascript | Vorteile assoziativer Arrays *Arrays in Javascript:*

Einfaches Durchlaufen der Eigenschaften  
**und Funktionen!**

```
pizza = {  
    name : "Diabolo",  
    preis : 7.5,  
    getPreis : function(){ return this.preis; }  
};  
var output = "";  
for ( x in pizza ){  
    output += x + " || " + pizza[x] + "\n";  
}  
alert(output);  
/*  
name || Diabolo  
preis || 7.5  
getPreis || function(){ return this.preis; }  
*/
```

Regular:  
var myCars=new Array();  
myCars[0]="Saab";  
myCars[1]="Volvo";  
myCars[2]="BMW";

Condensed:  
var myCars=new  
Array("Saab", "Volvo", "BMW");

Literal:  
var myCars=["Saab", "Volvo", "BMW"];

Siehe [http://www.w3schools.com/js/js\\_obj\\_array.asp](http://www.w3schools.com/js/js_obj_array.asp)

# JavaScript | Unobtrusive

## Grundsätze

- Abgrenzung zwischen **Verhalten**(Behavior), **Präsentation** (Presentation) und **Inhalt** (Content)  
*Beispiel:* Event Handling mit und ohne Event-Attribut in HTML
- Site soll auch ohne JavaScript funktionieren (☺)
- Mit JavaScript soll nur die Benutzerfreundlichkeit erhöht werden
- "Never trust JavaScript" – also keine kritischen Operationen damit durchführen ...

## URL

- Siehe: [https://www.w3.org/wiki/The\\_principles\\_of\\_unobtrusive\\_JavaScript](https://www.w3.org/wiki/The_principles_of_unobtrusive_JavaScript)

# Javascript | APIs & Bibliotheken

## APIs

- **List:** <https://developer.mozilla.org/en-US/docs/Web/API>
- AJAX & Fetch
- Webworker & Service Worker
- Canvas (Animation)
- WebSocket
- WebGL

## Bibliotheken

- **List:** [https://en.wikipedia.org/wiki/List\\_of\\_JavaScript\\_libraries](https://en.wikipedia.org/wiki/List_of_JavaScript_libraries)
- jQuery
- Chart.js
- D3.js

Welche API kennen Sie bereits?

Welche Bibliothek verwenden Sie am meisten?



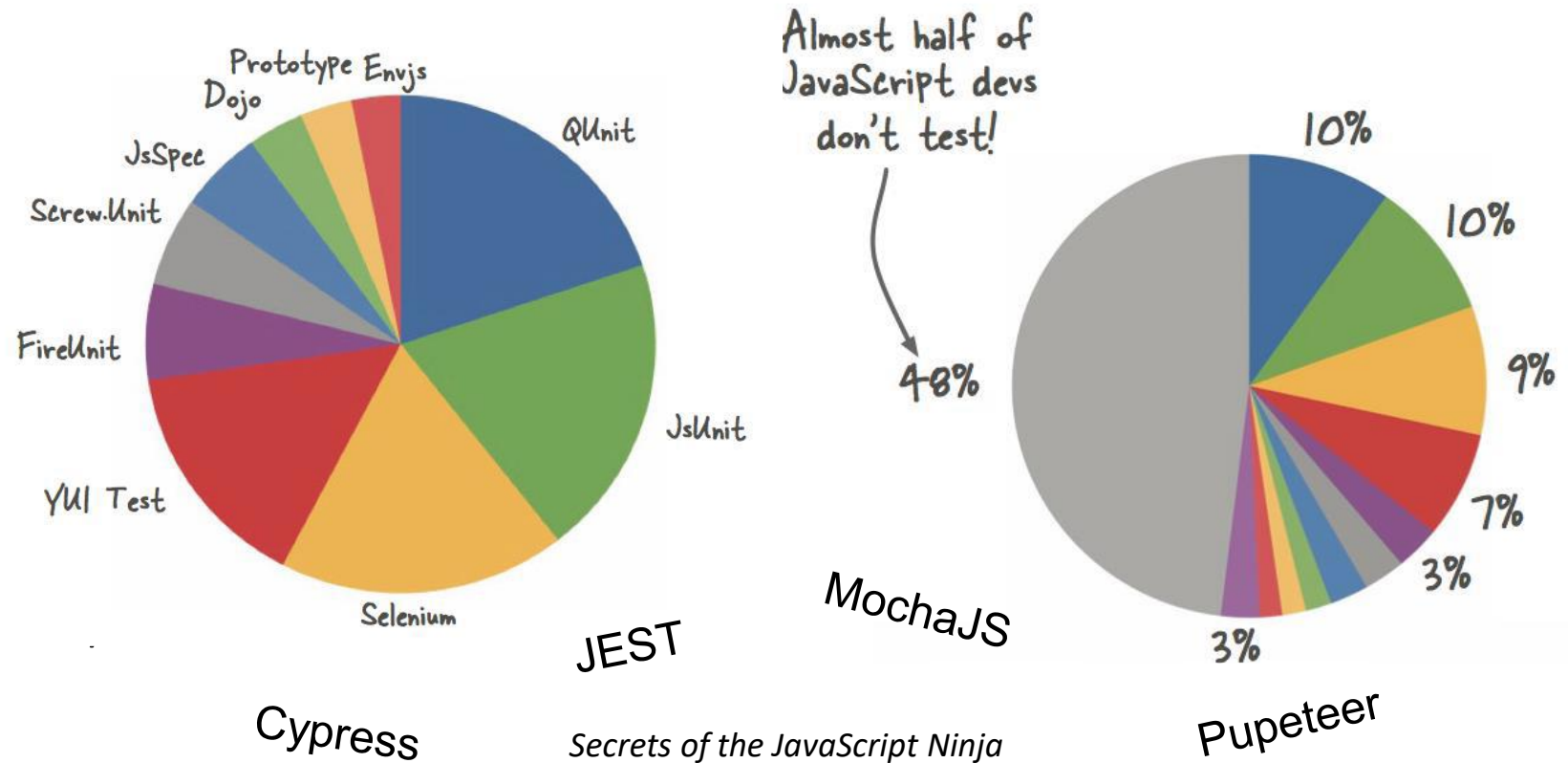
# JavaScript | Werkzeuge

 Logging



 Debugging

 Testing

- <https://jsfiddle.net/>
- <https://qunitjs.com/>
- <https://www.selenium.dev/>
- <https://clarle.github.io/yui3/y>



# Modernes Web Design

-  Responsives Web Design (RWD)
  - Reaktionsfähiges Layout
  - Der Aufbau eines Webauftritts wird für alle erdenklichen Fälle optimiert
  - Ziel ist für jeden Bildschirm und jedes Gerät einen fließenden Übergang zu ermöglichen
  - Rein clientseitig
  
-  Adaptives Web Design (AWD)
  - Bietet verschiedene Lösungen für bestimmte Bildschirme und Geräte
  - Maßgeschneidertes Layout (nicht eines für alle Eventualitäten sondern eines für genau diese Eventualität)
  - Serverseitige Funktionalitäten notwendig

# Single Page Application (SPA)

## Definition

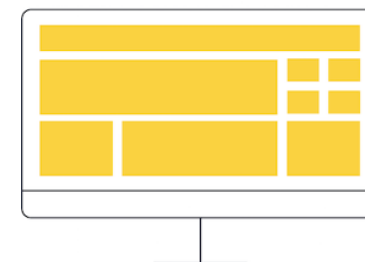
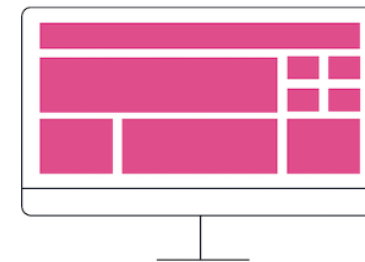
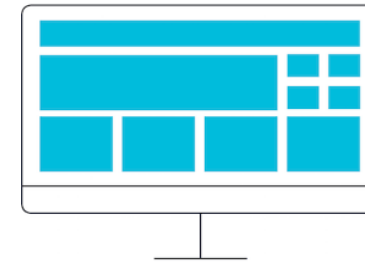
“An SPA (Single-page application) is a web app implementation that loads only a single web document, and then updates the body content of that single document via JavaScript APIs such as XMLHttpRequest and Fetch when different content is to be shown.”

## URLs

- <https://www.awwwards.com/websites/single-page/>
- <https://www.bloomreach.com/en/blog/2018/07/what-is-a-single-page-application.html> (Bild rechts)

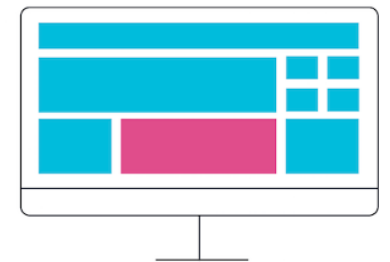
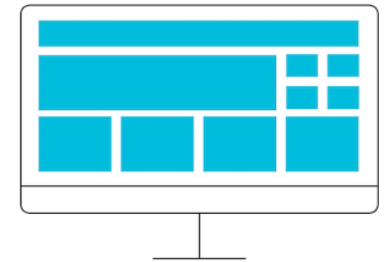
## Traditional

Every request for new information gives you a new version of the whole page.



## SPA

You request just the pieces you need.



# Single Page Application (SPA) | Vor- und Nachteile

## Vorteile

- Nur einmal wird eine Datei geladen
- Benutzerfreundlichkeit (Antwortzeiten, Dynamik, etc.)
- Entwicklung/Debugging ist einfacher
- Backend kann auch für andere Apps verwendet werden
- Effektives Cachen in beliebigem lokalen Speicher

## Nachteile

- Initiales Laden dauert
- Suchmaschinenoptimierung (SEO)  
*Wenig eindeutige Links*  
*Semantik schwer analysierbar*
- Analysen
- Geteilte Links zu Bereichen der App schwierig
- Weniger Sicherheit, weil viel Funktion am Client ist

Siehe auch

<https://www.bloomreach.com/en/blog/2018/07/what-is-a-single-page-application.html#single-page-applications-advantages>

# Single Page Application (SPA) | Frameworks

## 🌐 Frameworks

- VanillaJS
- Angular
- React
- Vue
- Aurelia



## 🌐 Komponentenbasierte Web-Entwicklung

- Fokus auf wiederverwendbare Komponenten
- Komponenten haben Eltern und Kinder
- Daten werden von Eltern an Kinder weitergereicht

HEADER



IMAGE SECTION



FOOTER



BLOG POST



FEATURES



TEXT



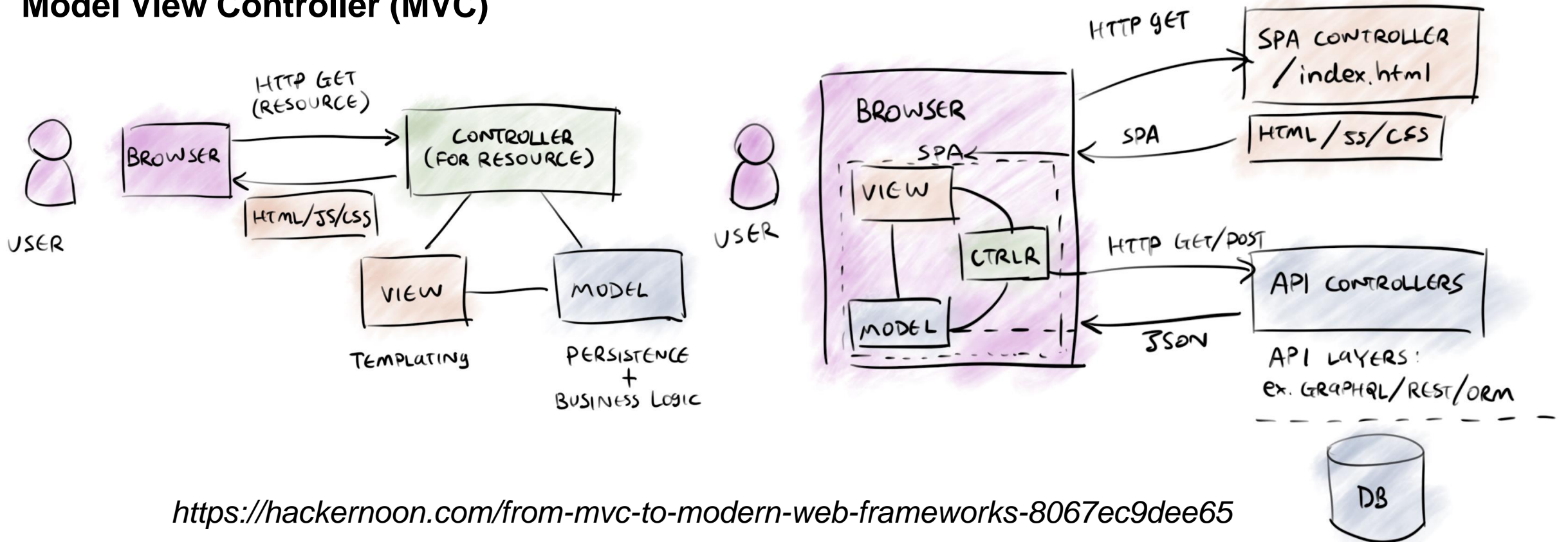
SIDE BLOCKS



<https://www.droptica.com/blog/component-based-design/>

# Single Page Application (SPA) | Moderne Web Frameworks

## Model View Controller (MVC)



<https://hackernoon.com/from-mvc-to-modern-web-frameworks-8067ec9dee65>



# Asynchronous JavaScript and XML (AJaX)

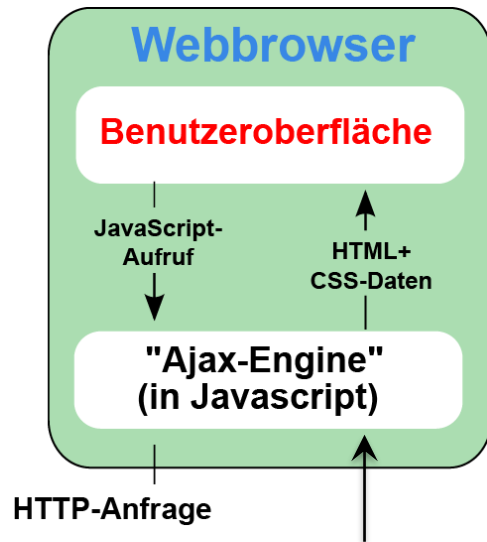
# Asynchronous JavaScript and XML

- 🌐 **Asynchrone Datenübertragung** zwischen Server und Browser
- 🌐 Seite **muss nicht neu geladen** werden
- 🌐 Schlüsseltechnik für Web2.0
- 🌐 Kombination von Funktionalitäten mehrerer Technologien
  - **(X)HTML** (und CSS) zur Darstellung auf der Webseite
  - **DOM** zur dynamischen Repräsentation und Interaktion der Daten / Inhalte im Dokument
  - **XMLHttpRequest** zur asynchronen Kommunikation mit dem Webserver
  - **JavaScript** als Schnittstelle zwischen diesen Technologien
- 🌐 **XHR - XmlHttpRequest**
  - Das XMLHttpRequest-Objekt wird zum Austausch von Daten zwischen Nutzer und Server verwendet.
  - Man kann damit ...
    - eine Webseite aktualisieren, ohne sie neu laden zu müssen
    - Daten vom Server anfordern, nachdem die Seite geladen ist
    - Daten zu einem Server im Hintergrund senden

[https://www.w3schools.com/js/js\\_ajax\\_intro.asp](https://www.w3schools.com/js/js_ajax_intro.asp)



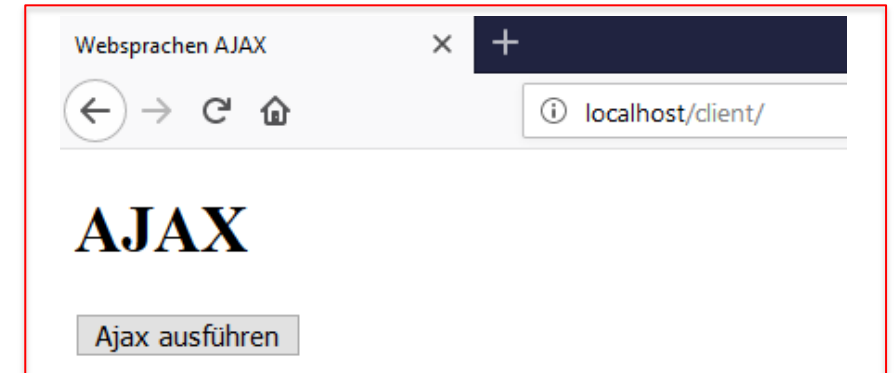
# AJaX - Ablauf | Benutzeroberfläche



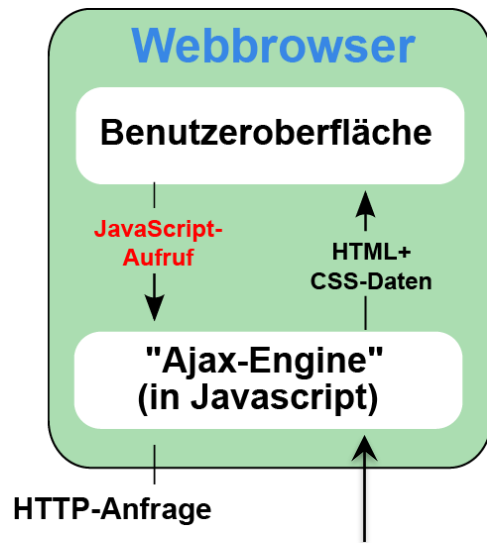
```

index.html
<!DOCTYPE html>
<html>
  <head>
    <title>Websprachen AJAX</title>
    <script src="js/ajax.js"></script>
  </head>
  <body>
    <h1>AJAX</h1>
    <button onclick="doAjax()">
      Ajax ausführen
    </button>
    <div id="result"></div>
  </body>
</html>

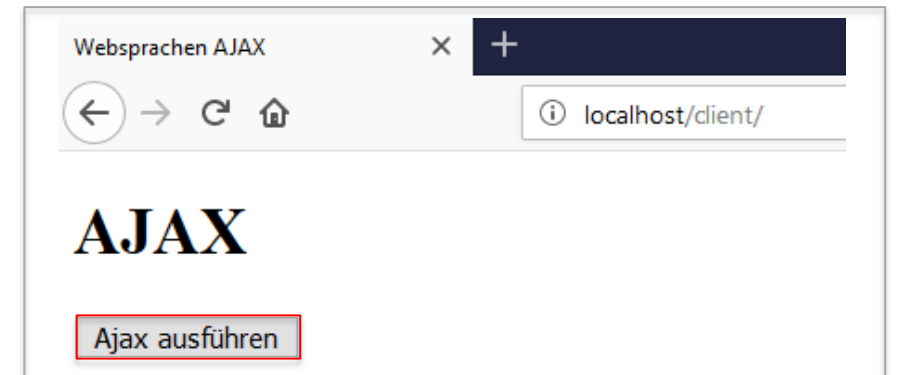
```



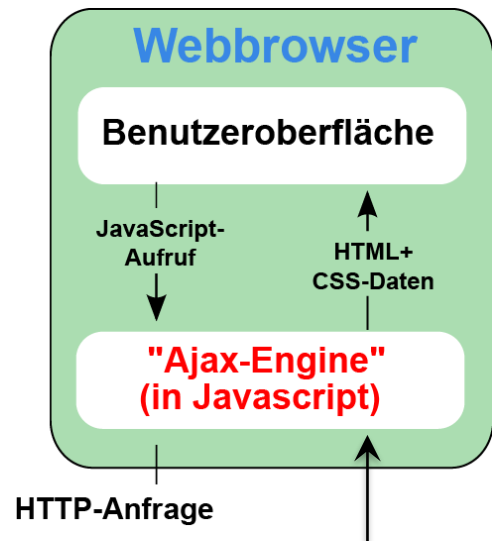
# AjAX - Ablauf | JavaScript aufrufen



```
index.html
<!DOCTYPE html>
<html>
  <head>
    <title>Websprachen AJAX</title>
    <script src="js/ajax.js"></script>
  </head>
  <body>
    <h1>AJAX</h1>
    <button onclick="doAjax()">
      Ajax ausführen
    </button>
    <div id="result"></div>
  </body>
</html>
```



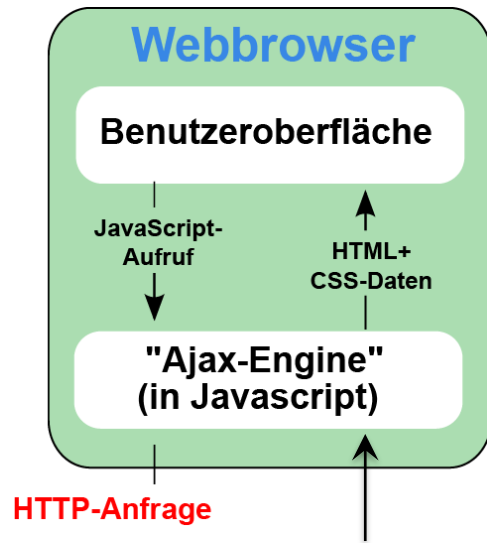
# AJaX - Ablauf | AJAX Engine



*ajax.js*

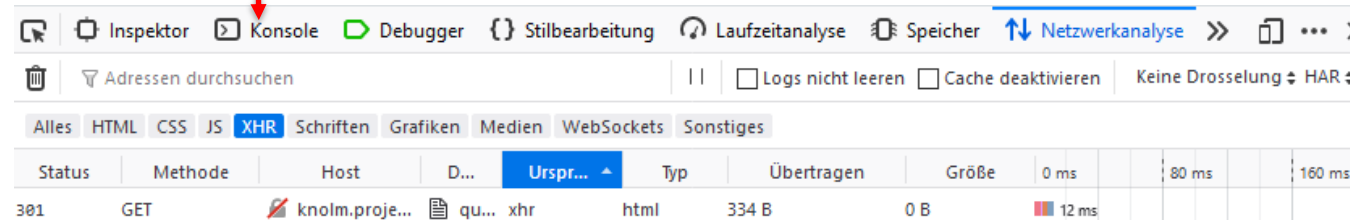
```
function doAjax() {
    let ajaxObj = new XMLHttpRequest();
    ajaxObj.open(
        "GET",
        "quotes.php");
    ajaxObj.onreadystatechange = function () {
        if (ajaxObj.readyState==4 && ajaxObj.status==200) {
            let elemResult = document.getElementById('result');
            elemResult.innerHTML = '';
            let quotesObj = JSON.parse(ajaxObj.responseText);
            quotesObj.quotes.forEach(function (item) {
                elemResult.innerHTML += item.text;
            });
        }
    };
    ajaxObj.send(null);
}
```

# AJaX - Ablauf | HTTP Anfrage



```

ajax.js
function doAjax() {
  let ajaxObj = new XMLHttpRequest();
  ajaxObj.open(
    "GET",
    "http://knolm.projekt-itm.fh-joanneum.at/api/quotes");
  ajaxObj.onreadystatechange = function () {
    if (ajaxObj.readyState==4 && ajaxObj.status==200) {
      let elemResult = document.getElementById('result');
      elemResult.innerHTML = '';
      let quotesObj = JSON.parse(ajaxObj.responseText);
      quotesObj.quotes.forEach(function (item) {
        elemResult.innerHTML += item+'<br/>';
      });
    }
  };
  ajaxObj.send(null);
}
  
```

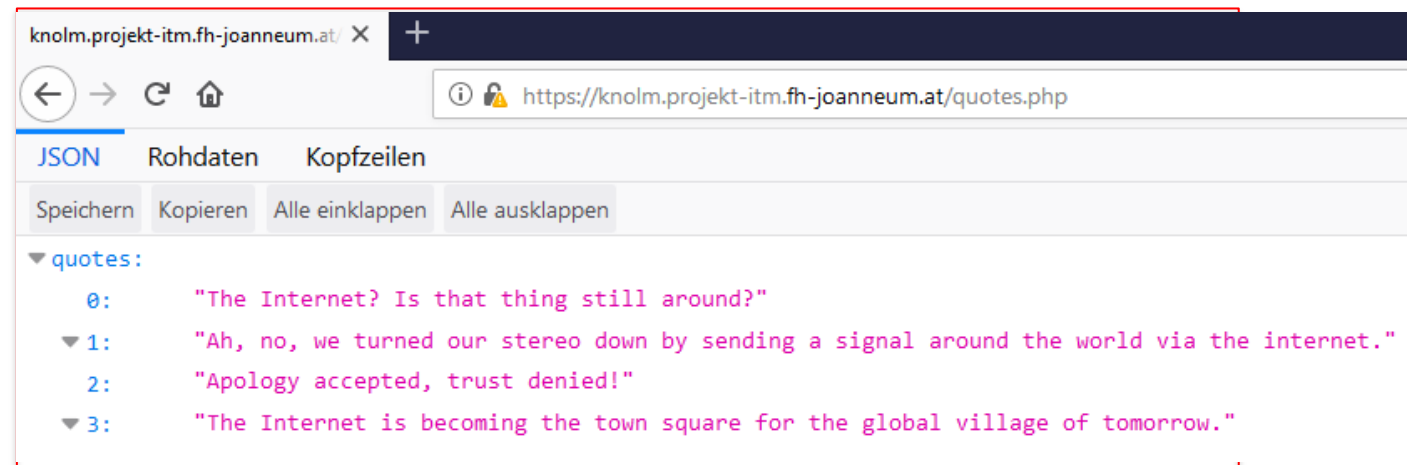
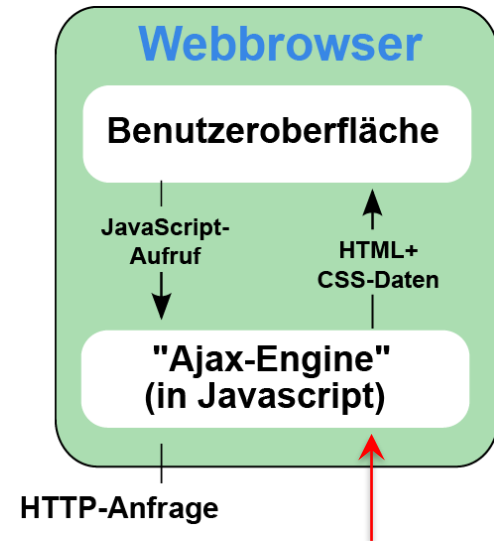


Status	Methode	Host	D...	Urspr...	Typ	Übertragen	Größe	0 ms	80 ms	160 ms
301	GET	knolm.proje...	qu...	xhr	html	334 B	0 B	12 ms		

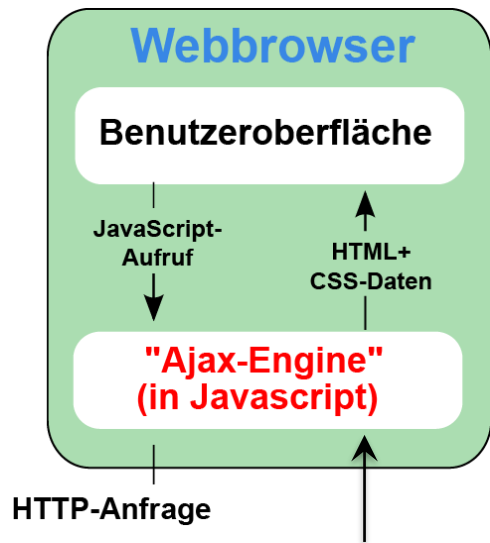
# AJaX - Ablauf | HTTP Antwort

## HTTP Response im JSON Format

```
{  
  "quotes": [  
    "The Internet? Is that thing still around?",  
    "Ah, no, we turned our stereo down by sending a signal  
    around the world via the internet.",  
    "Apology accepted, trust denied!",  
    "The Internet is becoming the town square for the  
    global village of tomorrow."  
  ]  
}
```



# AJAX - Ablauf | AJAX Engine & Anzeige im DOM



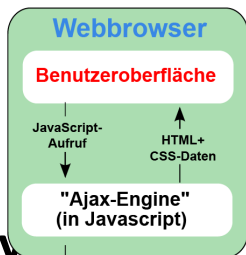
*ajax.js*

```
function doAjax() {
  let ajaxObj = new XMLHttpRequest();
  ajaxObj.open(
    "GET",
    "http://knolm.projekt-itm.fh-joanneum.at/quotes.php");
  ajaxObj.onreadystatechange = function () {
    let elemResult = document.getElementById('result');
    elemResult.innerHTML = '';
    let quotesObj = JSON.parse(ajaxObj.responseText);
    quotesObj.quotes.forEach(function (item) {
      elemResult.innerHTML += +item+'<br/>';
    });
  };
  ajaxObj.send(null);
}
```

## AJAX

Ajax ausführen

The Internet? Is that thing still around?  
Ah, no, we turned our stereo down by sending a signal around the world via t  
Apology accepted, trust denied!  
The Internet is becoming the town square for the global village of tomorrow.



# XHR Methode | `open()`

Diese Methode legt die Anfrage-Methode und -URL fest. Außerdem legt sie Anfrage-Benutzername und –Kennwort fest, und ob die Anfrage asynchron abgearbeitet wird.

## Argumente:

- 🌐 **method** die zu verwendende **HTTP Methode**, wie "GET", "POST", "PUT", "DELETE", etc.
- 🌐 **url**: URL, an den die Anfrage geschickt werden soll.
- 🌐 **async**: (*optional*) boole'scher Parameter
  - **true**: (Standardwert) gibt an, das Operation **asynchron** ausgeführt werden soll. Der Programmablauf wird nicht blockiert und die Benachrichtigung über die vollendete Transaktion erfolgt mittels Events.
  - **false blockiert** an der `send()`-Methode, bis die Antwort vollständig empfangen worden ist.
- 🌐 **user**: (*optional*) Benutzername zum Zweck der **Authentisierung**; ohne Angabe ist dies ein leerer String.
- 🌐 **password** (*optional*) zum Zweck der **Authentisierung**; ohne Angabe ist dies ein leerer String.

## Beispiel:

HTTP GET und Aufruf einer Datei im gleichen Verzeichnis:


```
ajaxObj.open( "GET", "quotes.php" );
```

```
void function open(  
    string method,  
    string url  
    [, boolean async = true  
    [, string user = null  
    [, string password = null]])
```

## XHR Methode | `setRequestHeader()`

Diese Methode setzt bestimmte Eigenschaften für die Ressourcen-Anfrage. Wenn der gleiche Header mehrmals gesetzt wird, werden die Werte kombiniert

### Argumente:

 **header:** HTTP-Header-Feldname

 **value:** Wert für das Header-Feld

```
void function setRequestHeader(  
    string header,  
    string value  
)
```

### Beispiele:

 Setzt den Inhaltstyp auf **JSON**:

```
ajaxObj.setRequestHeader('Content-Type', 'application/json');
```

 Setzt den Inhaltstyp auf **URL-kodierte Formulare Daten**:






```
ajaxObj.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
```



## XHR Eigenschaft | **readyState**

Diese Eigenschaft gibt über den Zustand des XHR Objekts Auskunft. Wenn sich diese Eigenschaft im Verlauf einer AJAX Abfrage ändert, wird ein "readystatechange"-Event ausgelöst.

### Zustände:

-  **UNSENT = 0**  
Objekt erstellt, open() wurde noch nicht aufgerufen
-  **OPENED = 1**  
open() wurde aufgerufen
-  **HEADERS\_RECEIVED = 2**  
send() wurde aufgerufen. Header und Status sind verfügbar
-  **LOADING = 3**  
Die Antwort vom Server wird gerade heruntergeladen
-  **DONE = 4**  
Die Operation ist fertig

## XHR Eigenschaft | `onreadystatechange`

🌐 Hinter dieser Eigenschaft verbirgt sich ein EventHandler. Mit jeder Statusänderung (`readyState`) wird diese Callback-Funktion aufgerufen

🌐 **Beispiel:**

```
ajaxObj.onreadystatechange = function () {  
    console.log(  
        "readyState=" + getState(ajaxObj.readyState) +  
        "; status=" + ajaxObj.status +  
        '(' + ajaxObj.statusText + ')');  
};
```

## XHR Methode | **send()**

Diese Methode schickt die Anfrage los.

### **Argument:**

**data:** (optional) Daten, die im Körper der Anfrage gesendet werden.

### **Achtung:**

Damit der Webserver die Daten korrekt interpretiert, muss man den "**Content-Type**" mittels "setRequestHeaders()" angeben!

```
void function send(  
    [data=null]  
)
```

# XHR Alternative | window.fetch

Fetch abstrahiert die Hauptkomponenten von HTTP

Verwendet Promises!

```
function doFetch() {
  fetch('quotes.php')
    .then(function(response) {
      if (response.ok)
        return response.json();
      else
        throw new Error('Zitate konnten nicht geladen werden');
    })
    .then(function(json) {
      let elemResult = document.getElementById('result');
      elemResult.innerHTML = '<b>Quotes:</b><br />';
      json.quotes.forEach(function (item) {
        elemResult.innerHTML += item+'<br/>';
      });
    })
    .catch(function(err) {
      console.error("Fehler", err);
    });
}
```

[https://www.w3schools.com/js/js\\_api\\_fetch.asp](https://www.w3schools.com/js/js_api_fetch.asp)

# Promises

- 🌐 Ein Promise is `new Promise(function(resolve, reject) { ... });` icken API kapselt.
  - 🌐 Beim Erzeugen wird eine Funktion als Parameter übergeben, der so genannte **Exekutor**
  - 🌐 Dieser bekommt seinerseits **zwei Callback-Funktionen** als Parameter
    - den Erfüller (**resolve**) und
    - den Zurückweiser (**reject**).
  - 🌐 Erfüller und Zurückweiser erwarten **jeweils einen Parameter (Erfüllungswert und Zurückweisungsgrund)**, die vom Promise später zur Verfügung gestellt werden
  - 🌐 **Zustände:**
    - **Pending** (Wartend)
    - **Fulfilled** (Erfüllt)
    - **Rejected** (Abgelehnt)
- [https://www.w3schools.com/js/js\\_promise.asp](https://www.w3schools.com/js/js_promise.asp)

# Promises | Callbacks & Beispiel

## 🌐 Methode: **then(resolve, reject)**

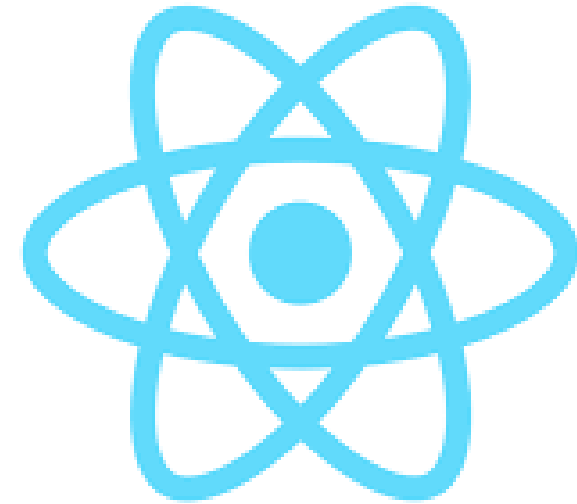
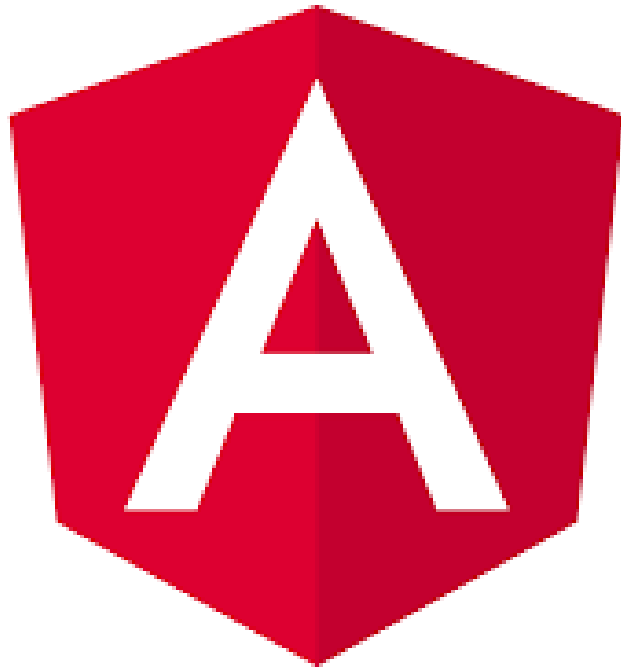
- Kann selbst aber auch wieder ein **Promise Objekt zurückgeben!**  
(Kettenbildung...)

## 🌐 Methode: **catch(error)**

- Wird zur Fehlerbehandlung zurückgegeben

```
function asyncFunction(test) {
    const promise = new Promise(
        (resolve, reject) => {
            if(test){
                resolve("OK");
            }else{
                reject("NOK");
            }
        }
    );
    return promise;
}

function callAsyncFunction(test) {
    asyncFunction(test).then(function (data_resolve) {
        alert(data_resolve)
    }, function (data_reject) {
        alert(data_reject)
    });
}
```



# JS&TS Frameworks

# Web Frameworks

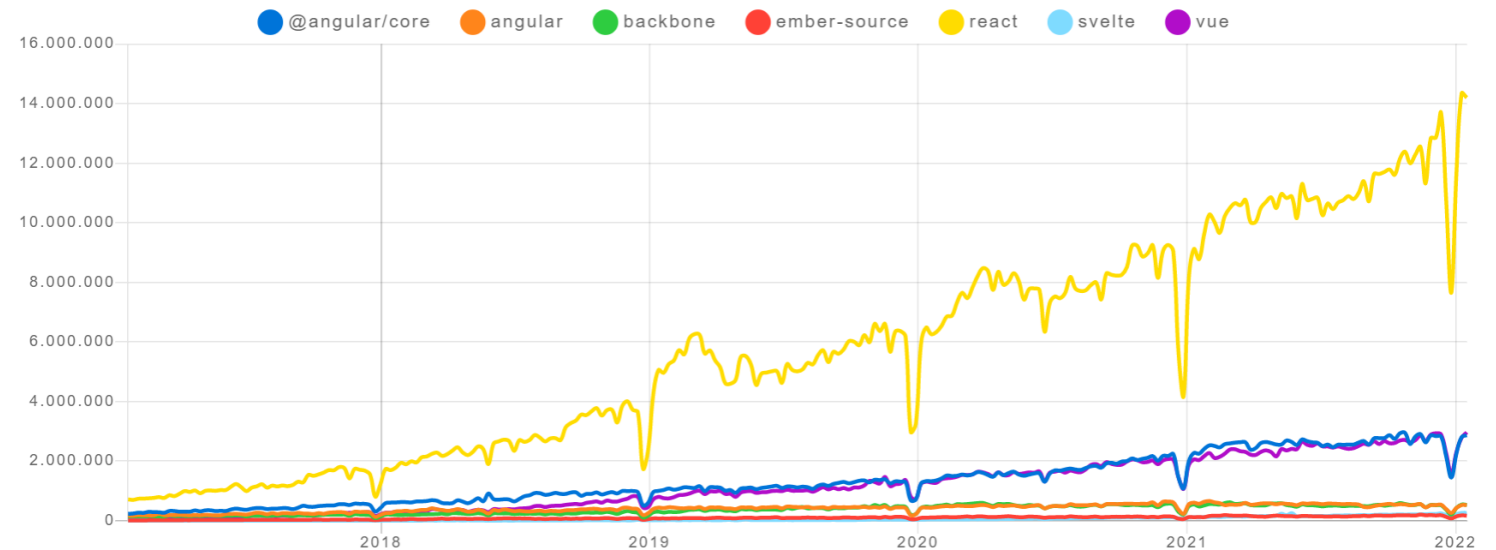
## 🌐 Kernfunktionen

- DOM Management -
- Data Binding
- Rendering

## 🌐 Termini

- Komponenten und Templating
- Direktiven
- Routing
- Reaktive Architektur (RxJS)
- Testing
- Modularisierung

Downloads in past 5 Years ▾































<https://www.npmtrends.com/@angular/core-vs-angular-vs-backbone-vs-ember-source-vs-react-vs-svelte-vs-vue>



# npm-Trends zu Web Frameworks 2

## Stats

				Stars	Issues	Version	Updated <sup>?</sup>	Created <sup>?</sup>	Size
	@angular/core	  	-	-	13.2.0	17 hours ago	6 years ago	minzipped size 73.4 KB	
	angular	  	59.588	464	1.8.2	a year ago	10 years ago	minzipped size 62.3 KB	
	backbone	  	27.813	96	1.4.0	3 years ago	11 years ago	bundlephobia 429	
	ember-source	  	22.131	432	4.1.0	a month ago	5 years ago	minzipped size 401.6 KB	
	react	  	181.297	920	17.0.2	10 months ago	10 years ago	minzipped size 2.8 KB	
	svelte	  	55.189	642	3.46.3	21 hours ago	5 years ago	minzipped size 1.6 KB	
	vue	  	192.537	553	2.6.14	8 months ago	8 years ago	minzipped size 22.9 KB	



## Angular

<https://angular.io/>

- 🌐 Startk komponentenbasiert
- 🌐 Clientseitiger Framework für SPAs
- 🌐 Tendenz zur Plattform (z.B. mit API)
- 🌐 Klare Vorgaben zur Struktur
- 🌐 Fokus auf
  - Dependency Injection
  - Test Driven Development



## React

<https://reactjs.org/>

- 📦 **Minimal komponentenbasiert, was eine eigene Struktur ermöglicht/erfordert**
- 📦 **Viele Module bieten ein breites Spektrum für Lösungen**
- 📦 **Höherer Aufwand an Integration & Wartung**

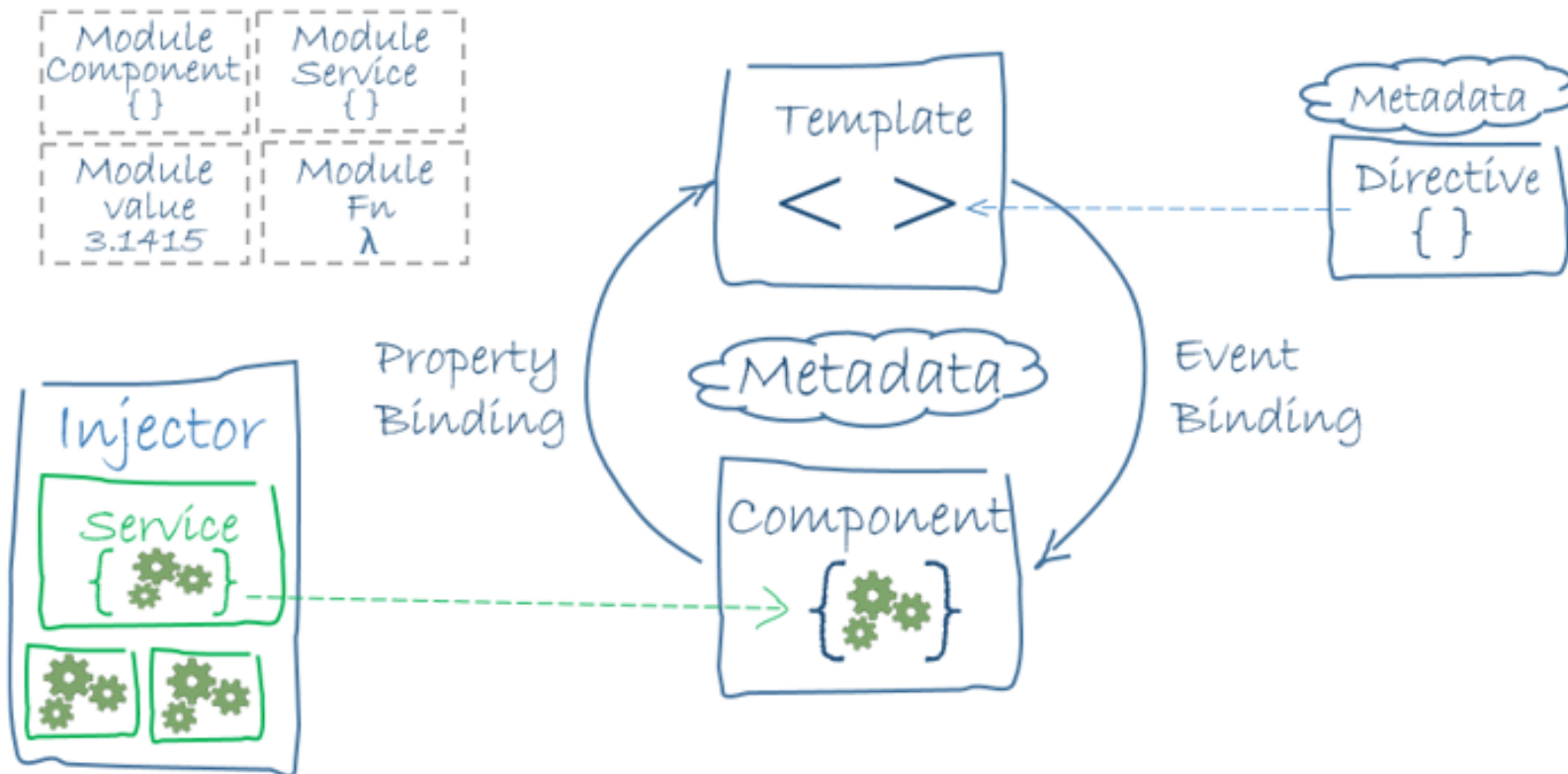


## Vue

<https://vuejs.org/>

- 📦 **Zwischen React & Angular**

# Angular | Architektur

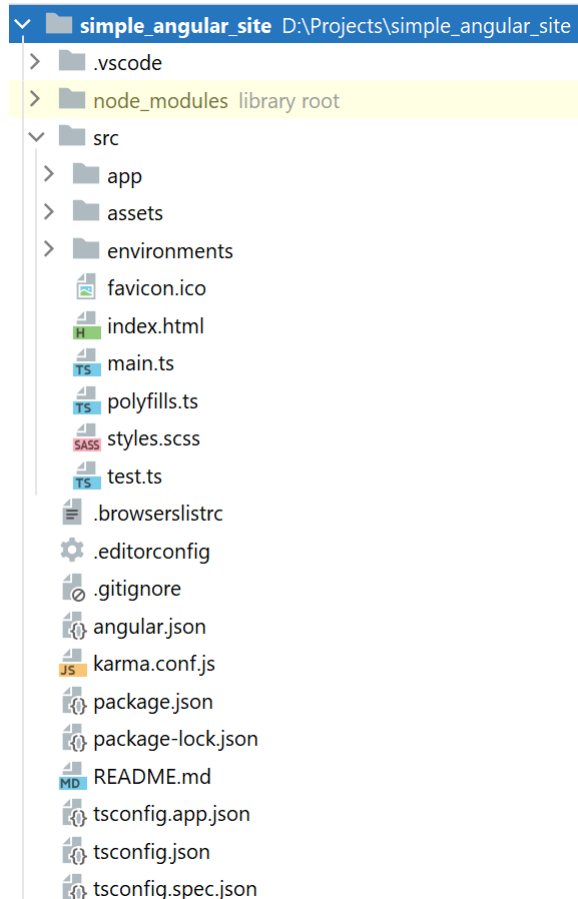


- 🌐 **Module ...**
  - ... bieten verschiedene Funktionen
  - ... werden nach Bedarf geladen
  - ... haben ein "**Root Module**", mit welchem die **App startet ('AppModule')**
- 🌐 **Komponenten ...**
  - ... bestehen aus Modulen
  - ... definieren Views (Templates)
  - ... benutzen Dienste (Services)
- 🌐 **Templates**
  - ... kombinieren HTML & Angular Markup
  - ... beinhalten Direktiven für Logik
  - ... beinhalten Bindings, um Daten der App mit dem DOM zu verbinden
- 🌐 **Services & Dependency Injection**
  - ... helfen Daten zwischen Views zu teilen
  - ... helfen beim Routing (Navigation)

# Angular | Erste Schritte

- 🌐 `npm install -g @angular/cli`
  - Angular Command Line Interface global installieren
- 🌐 `ng new simple_angular_site; cd simple_angular_site`
  - Damit wird ihr Angular Projekt erstellt
  - Dialogoptionen fragen nach Routing<sub>ja</sub> und dem gescipteten CSS Format<sub>scss</sub>
- 🌐 `ng serve --open`
  - Mit diesem Befehl wird ein Server und die Überwachung von Dateiänderungen gestartet
  - Unter `http://localhost:4200/` ist die App verfügbar
- 🌐 **Tour of Heroes**
  - Tutorial unter `https://angular.io/tutorial`
  - Bringt alle erwähnten Konzepte näher ...
- 🌐 **Online Testen**
  - Über die URL `https://stackblitz.com/` hat man ein komplette Angular Projekt als Spielwiese zur Verfügung

# Angular | Projektstruktur



- **src**
    - app ... Komponente
    - assets ... Bilder und Dateien als Ressource
    - environments ... Konfigurationen zum Erstellen z.B.: Produktiv oder Test
    - main.ts ... Haupteinstiegspunkt über Modul "AppModule"
    - styles.scss ... Gekriptetes CSS für das Projekt
    - test.ts ... Haupteinstiegspunkt für das Teste
  - **angular.json**
    - CLI Konfiguration für die Angular Arbeitsumgebung
  - **karma.conf.js**
    - Basis für Testing
    - Siehe <http://karma-runner.github.io/6.3/index.html>
  - **tsconfig.json**
    - Typescript Konfiguration
  - **Projektabhängig können die Inhalte variieren....**
    - Zu beachten: Sie haben ein NodeJS Projekt mitsamt den üblichen Inhalten (node\_modules, package.json,...)

# Angular | Module

- 🌐 Eine App startet mit dem 'AppModule'
- 🌐 Erstellen: `ng generate module <module_name>`
- 🌐 Klasse mit dem `@NgModule` Decorator
- 🌐 Beispiel für verfügbare Module mit eigenen APIs:
  - FormsModule
  - ReactiveFormsModule
  - CommonsModule
  - HttpClientModule
  - RouterModule
- 🌐 Best Practices
  - Modularisierung: Funktion kapseln
  - Klein bleiben → **Lazy Loading**

*app.module.ts*

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})

export class AppModule { }
```

*Komponenten, die zum Modul gehören*

*Andere Module, die benötigt werden z.B. für das Routing*

*Services auf Modulebene  
Start mit Hauptansicht (nur in Root Modul)*

# Angular | Services

- 🌐 In Verbindung mit "Dependency Injection" wird eine Klasse in die App eingefügt
- 🌐 Erstellen: `ng generate service <service_name>`
- 🌐 Klasse mit dem `@Injectable` Decorator
- 🌐 Varianten
  - Als "Singleton" (nur eine Instanz) mittels '**providedIn**'
  - **Multiple Instanzen** ohne diese Eigenschaft und mittels Angabe ('**providers**') in der Komponente (*siehe nächste Folie*)
- 🌐 Bsp.: Logger  
siehe <https://angular.io/guide/architecture-services>

```
test.service.ts
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
})

export class TestService {

  constructor() { }
}
```

# Angular | Komponenten

- 🌐 Komponenten bestehen aus
  - HTML Template `<component-name>.component.html`
  - Typescript Klasse `<component-name>.component.ts`
  - CSS-Selektor und -Stile `<component-name>.component.css`
  - Test Spezifikation `<component-name>.component.spec.ts`

🌐 Erstellen: `ng generate component <component-name>`

🌐 Klasse mit dem `@Component` Decorator

- **selector**: Komponente **einfügen**, wo im Template **entsprechender Tag** verwendet wird
- **templateUrl**: **Vorlage** relativ zum Modul
- **styleUrls**: **Gescriptete Stile** relativ zum Modul
- **providers**: Service(s) für die Komponente

*app.component.ts*

```
import { Component } from '@angular/core';
import { TestService } from './test.service';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss'],
  providers: [ TestService ]
})

export class AppComponent {
  title = 'simple_angular_site';
}
```

*index.html*

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>SimpleAngularSite</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```



# Angular | Templates (Vorlagen)

- 🌐 Text Interpolation
  - Eigene Strings einfach im Template integrieren: `{{ <variable> }}`
  - Beispiel: `<p> is the <i>interpolated</i> image.</p>`
  - Berechnungen, Funktionen, etc. können ebenso eingebunden werden.
  - siehe <https://angular.io/guide/interpolation>
- 🌐 Property Binding
  - Wird für Eigenschaften von Elementen verwendet
  - Beispiel: `<img [src]="itemImageUrl">`
  - Siehe <https://angular.io/guide/property-binding>
- 🌐 Template Statements
  - Werden für Events verwendet
  - Beispiel: `<button (click)="deleteHero()">Delete hero</button>`
  - Events werden dabei in geschwungenen Klammern definiert
- 🌐 Event Binding

## Data Binding Kategorien

"One-Way" von Datenquelle zu Ziel  
`{{Ausdruck}}`  
`[Ziel]="Ausdruck"`

"One-Way" von  
Ziel(View) zu Datenquelle  
`(Ziel)="Statement"`

"Two Way" – in beide  
Richtungen  
`[(Ziel)]="Ausdruck"`

# Angular | Routing

<https://angular.io/guide/router>

## Angular Router

## Erstellen einer App mit Routing:

```
ng new routing-app --routing --defaults
```

## Schritte für das Routing

- Importieren der Komponenten in 'AppRoutingModule'
- Routen in 'routes' hinzufügen (siehe Optionen)
- 'AppRoutingModule' in AppModule einbinden

## Optionen

- Pfad: **path und component** für Pfad und anzuzeigende Komponente
- Redirecting: **redirectTo und pathMatch**
- Wildcards: **\*\*** für keinen bestimmten Pfad
- Hierarchien: **children**, um Subpfade anzugeben!
- Parameter: **Pfad/<parameter>** und **ngOnInit** Interface

*app-routing.module.ts*

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';

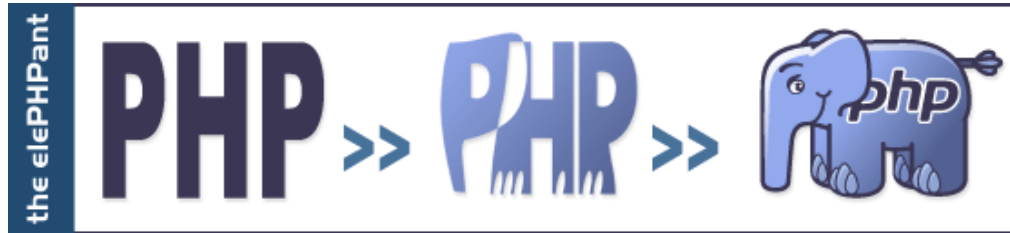
const routes: Routes = [];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Siehe <https://angular.io/guide/router>



# Pre Hypertext Processing (PHP) Serverseitiges Scripting



- 🌐 PHP steht für
  - **PHP: Hypertext Preprocessor** oder logischer verdreht
  - **Pre Hypertext Processing/or** und historisch
  - **Personal Homepage Tools (PHP1)**
- 🌐 Es ist als **serverseitige Skriptsprache** konzipiert
- 🌐 Scripts werden am Server **interpretiert**
- 🌐 Es muss **am Webserver installiert und konfiguriert** sein, um in Webseiten eingebunden zu werden
- 🌐 Die Homepage ist auf **<http://php.net>**

<https://www.w3schools.com/php/default.asp>

# Ausführung

- 🌐 Die Konfigurationsdatei für PHP nennt sich **php.ini**
- 🌐 Kernkompetenz ist die **serverseitige Scripting** für Webseiten und Webapplikationen
- 🌐 **PHP Scripts** in der Shell sind möglich  

```
#!/usr/bin/php  
echo "Hello World";
```
- 🌐 Möglich aber kaum verwendet ist die Verwendung als **Desktop-Anwendung** mittels PHP-GTK GUIs  
(**Gimp Toolkit** oder auch Graphical Toolkit)

# Grundlegende Syntax

🌐 Scriptblock innerhalb HTML

```
<?php /* Code */ ?>
```

🌐 Beispiel

```
<?php echo "Hello World"; ?>
```

🌐 Abschließen von Befehlen mit Strichpunkt ( ; )

🌐 Kommentare

○ Einzeilig

```
// Eine Zeile,
```

○ Mehrzeilig:

```
/* oder mehrere  
Zeilen lang */
```

# Variablen | Grundlagen

- 🌐 **Beginnen mit \$**
  - **Case Sensitive**
  - Name **beginnt mit Buchstaben oder \_**
  - **Alphanumerische Zeichen** und **\_** können verwendet werden
- 🌐 **Schwache Typisierung**
  - **Typ wird durch den zugewiesenen Wert bestimmt**
- 🌐 **Referenzen/Zeiger auf Variablen durch &**

[https://www.w3schools.com/php/php\\_variables.asp](https://www.w3schools.com/php/php_variables.asp)

<https://www.php.net/manual/de/language.variables.basics.php>

```
$var_name = "Mathias";  
$name = "Max";  
$nr = 12;
```

## Tip: Schnellausgabe

```
<?=$var_name?>
```

```
$ref_name = &$var_name;  
echo $ref_name
```

**Ausgabe: Mathias**

# Variablen | Standardtypen

 **Boolean** `<?php $isTrue = True; // oder False`

 **Integer** `<?php $iNumber = 20; // oder -20  
oder 024 dezimal  
oder 0x14 oktal  
oder 0b10100 hexadez.  
binär`

 **Float** `<?php $fNumber = 1.2; // oder 1.2e1 Exponential`

 **String** `<?php $sHello = 'Hello'; // oder "Hello"`

- Bei einfachen Hochkommas (') funktionieren **Maskierungen** und **Escapesequenzen NICHT**
- Bei doppelten Hochkommas (") funktionieren **Maskierungen** und **Escapesequenzen**

[https://www.w3schools.com/php/php\\_datatypes.asp](https://www.w3schools.com/php/php_datatypes.asp)  
<https://www.php.net/manual/de/language.types.php>

## Alternative „heredoc“

```
$bar = <<<MEINTEXTHIER  
$sHello World  
MEINTEXTHIER;  
Ausgabe: Hello World
```

## Alternative „nowdoc“

```
$bar = <<<'OHNEESCAPING'  
$sHello World  
OHNEESCAPING;  
Ausgabe: $sHello World
```

*ähnlich wie  
in Linux*



# Variablen | Spezialtypen

- 🌐 **Variable ohne Wert**  
Wenn der Variable **nichts oder explizit `Null`** zugewiesen oder mit `unset($value)` gelöscht wurde
- 🌐 **Not a Number**  
Wenn das Ergebnis einer Berechnung **nicht repräsentierbar** oder **undefiniert** ist
- 🌐 **Resource** (siehe <http://php.net/manual/de/resource.php>)
- 🌐 **Welcher Typ?** Mit der Funktion `gettype` ( `mixed $var` ) kann man es herausfinden!

## **NULL**


```
$value = NULL; is_null($value);
```

## **NaN**

```
$value = NaN; is_nan($value);
```

# Variablen | Gültigkeit

 **Lokal**  
Wenn innerhalb von Funktionen definiert und gültig

 **Global**  
Außerhalb von Funktionen definiert  
Kann **in** Funktionen nur über Schlüsselwort verwendet werden: **global**  
**\$varGlobal**;  
Werden im Array **\$GLOBALS** gespeichert!

 **Statisch**  
Damit geht ein Wert in einer Funktion nicht verloren sondern bleibt bis zum nächsten Funktionsaufruf **static** **\$varStatic**;

# Variablen | Gültigkeit - Beispiel

```
<?php
```

```
$varGlobal = "Statische Variable: ";  
function funktionTest(){  
    global $varGlobal;  
    static $varStatic = 0;  
    $sUmbruch = "<br>"; // Lokal  
    $varStatic++;  
    echo $varGlobal . $varStatic . $sUmbruch;  
}  
funktionTest();  
funktionTest();
```

Ausgabe:

```
Statische Variable: 1  
Statische Variable: 2
```

*Wichtig!  
In Javascript ist die  
Gültigkeit etwas anders!*

# Arrays

Ist eigentlich eine **geordnete Zuweisung oder Ordered Map**

```
array (  
  Schlüssel => Wert,  
  Schlüssel => Wert  
)
```

**Schlüssel:** String oder Integer

**Wert:** Jeder Typ

*Besondere Variablen  
presenting ...*

[https://www.w3schools.com/php/php\\_arrays.asp](https://www.w3schools.com/php/php_arrays.asp)  
<https://www.php.net/manual/de/book.array.php>

## Arrays | Beispiele

```
<?php $arrTest01 = array(1,2,3,4); ?>
```

```
<?php $arrTest02 =  
    array( 1=>"Eins", 2=>"Zwei"  
); ?>
```

```
<?php $arrTest03 =  
    array("alphabet" =>  
        array("a"=>1, "b"=>2)  
); ?>
```

```
<?php var_dump($arrTest03); // Anzeigen
```

## Arrays | Zugriff

```
echo $arrTest01[0]; // Ordinate
```

Ausgabe: 1

```
echo $arrTest02[1]; // Index
```

Ausgabe: Eins

```
echo $arrTest03["alphabet"]["a"];
```

Ausgabe: 1

```
foreach($arrTest03 as $index => $wert) {  
    echo $index . " " . $wert["a"];  
}
```

Ausgabe: alphabet 1

```
array(1,2,3,4);  
array( 1=>"Eins", 2=>"Zwei");
```

```
array("alphabet" =>array("a"=>1,"b"=>2))
```

## Arrays | Bearbeitung

**Hinzufügen ohne Ordinate mit []**  
(Ordinate wird automatisch angelegt)

```
array( 1=>"Eins", 2=>"Zwei");
```

**Hinzufügen mit Ordinate oder Zeichenkette**

```
$arrTest02[] = "Drei";
```

**Entfernen eines (unerwünschten) Elements oder ganzen Array leeren**

```
$arrTest02[3] = "Drei";
```

```
$arrTest02["xy"] = „test“;
```

```
unset($arrTest02["xy"]);
```

```
unset($arrTest02);
```

# Vordefinierte Variablen

*Alles Arrays!*

`$GLOBALS`

Referenziert alle Variablen,  
die im globalen Gültigkeitsbereich vorhanden sind

`$_SERVER`

Informationen über Server und Ausführungsumgebung

`$_GET`

HTTP GET-Variablen

`$_POST`

HTTP POST-Variablen

`$_FILES`

HTTP Dateiupload-Variablen

`$_REQUEST`

HTTP Request-Variablen

`$_SESSION`

Sessionvariablen

`$_ENV`

Umgebungsvariablen

*Eine Auswahl-  
mehr auf <http://php.net>*

`$_COOKIE`

HTTP Cookies

`$argc`

Die Anzahl der an das Skript übergebenen Argumente

Array der an das Skript übergebenen Argumente



## Variable Variablen(namen)

Mit einem weiteren `$` vor dem Variablennamen, wird auf den Inhalt der Variablen als Name referenziert

```
$meinvariablenname = "vorname";  
$$meinvariablenname = "Mathias"; // $vorname = "Mathias";  
echo "$meinvariablenname ${$meinvariablenname}";  
echo "$meinvariablenname $vorname";
```

Ausgabe in beiden Fällen: `vorname Mathias`

Anmerkung: Das lässt sich beliebig fortsetzen – zum Beispiel: `$$$$$variable`

*Falls man mal so einen Anwendungsfall hat ...*



# Konstanten

## Selbstdefiniert

Beispiel:

```
<?php define ("FOOBAR", "irgendwas");
```

[https://www.w3schools.com/php/php\\_constants.asp](https://www.w3schools.com/php/php_constants.asp)

## „Magische“ Konstanten vom System

Beispiele:

```
__LINE__ // Aktuelle Zeilennummer oder  
__DIR__ // Aktuelles Verzeichnis
```

<https://www.php.net/manual/de/language.constants.php>

<https://www.php.net/manual/de/language.constants.predefined.php>

Funktionen:

```
bool defined ( string $name );  
// ob eine Konstante definiert ist
```

# Funktionen

- 🌐 Funktionen werden mit dem **Schlüsselwort** „function“ definiert: **function** funktion () {  
...  
}
- 🌐 Es können **Parameter** und Referenzen/Zeiger übergeben werden: **function**  
**funktion (&\$a) { ...**
- 🌐 Es können **Vorgabeparamter** definiert werden:  
**function funktion (\$a, \$b="Baehh") {**  
... **//nur am Ende**
- 🌐 Eine variable Anzahl von Parametern wird auch unterstützt!

🌐 **Rückgabewerte** mit "return": **return \$x;**  
Standardmäßig (ohne **return**) wird **NULL** zurückgegeben

🌐 **Variablenfunktion**  
Wenn man an eine Variable () anhängt, versucht der Interpreter eine Funktion mit dem Namen des Variablenwertes aufzurufen

```
function test(){ echo "TEST"; }  
$meinefunktion = "test";  
$meinefunktion();
```

*Ausgabe: Test*

# Dateien einbinden

**include** bindet eine angegebene Datei ein und führt sie aus.

**require** ist ähnlich, wirft aber im Fehlerfall einen `E_COMPILE_ERROR` Fehler und beendet so die Programmausführung während `include` nur eine Warnung (`E_WARNING`) generiert.

**include\_once**, **require\_once** überprüft, ob die Datei schon eingebunden wurde.

## Beispiel

```
<html>
  <body>
    <?php include("header.php"); ?>
    <h1>Welcome to my home page!</h1>
    <p>Some text.</p>
  </body>
</html>
```

[https://www.w3schools.com/php/php\\_includes.asp](https://www.w3schools.com/php/php_includes.asp)

# Parameterübergabe

🌐 Übergabe von Parametern via HTTP GET und POST

🌐 Parameter über URL

```
http://localhost/welcome.php?name=Mathias  
echo $_GET["name"];
```

🌐 Parameter werden im „body“ eingeschlossen

```
<form action="welcome.php" method="post">  
  Name: <input type="text" name="name" />  
echo $_POST["name"];
```

[https://www.w3schools.com/php/php\\_forms.asp](https://www.w3schools.com/php/php_forms.asp)

# PHP Beispiel

```
.    welcome.html
<form action="welcome.php" method="post">
    Name: <input type="text" name="username" />
    Alter: <input type="text" name="age" />
    <input type="submit" />
</form>
```

```
.    welcome.php
```

```
.    Welcome <?php echo $_POST["username"]; ?>!<br/>
    You are <?php echo $_POST["age"]; ?> years old.
.
```



# Dateizugriff

🌐 Dateien als Speicher von Daten (auch XML)

🌐 Rechte beachten !!!

🌐 Auch entfernte Dateien/URLs können geladen werden:

```
$datei = fopen ("http://www.fh-joanneum.at/", "r");
```

🌐 Achtung bei gleichzeitigem Zugriff

🌐 **<?php**

```
    $file = fopen ("welcome.txt", "r");  
    // do something  
    fclose ($file);
```

**?>**

🌐 File Modes – siehe <http://at1.php.net/manual/de/function.fopen.php>

- r, r+, w, w+, a, a+, x, x+

# Sessions & Cookies

HTTP ist „stateless“

Oft ist aber eine Beobachtung des Zustands einer Webanwendung notwendig. Z.B.:

- Eingeloggt bleiben
- Online Einkauf - Warenkorb

Sessions (Variablen die den Zustand eines einzelnen Users speichern)

```
session_start(); $_SESSION['Sessionname']="Hallo";  
// Code  
echo "Name=" . $_SESSION['Sessionname'];
```

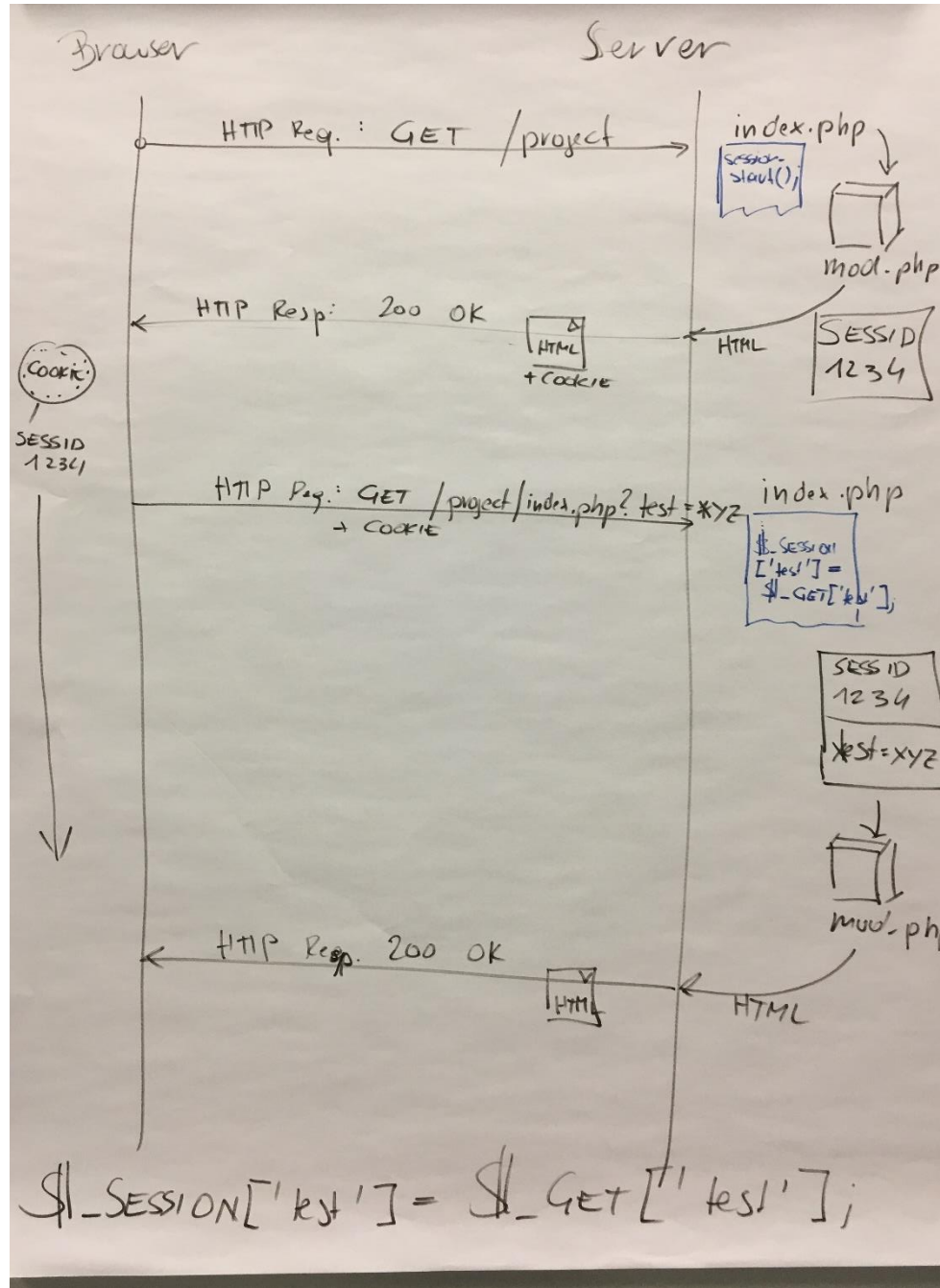
Cookies (Vom Server an den Client geschickte Strings dort gespeichert)

```
setcookie(name, value, expire, path, domain);  
// Code  
echo $_COOKIE["user"];
```

<https://www.php.net/manual/de/book.session.php>  
[https://www.w3schools.com/php/php\\_sessions.asp](https://www.w3schools.com/php/php_sessions.asp)  
[https://www.w3schools.com/php/php\\_cookies.asp](https://www.w3schools.com/php/php_cookies.asp)



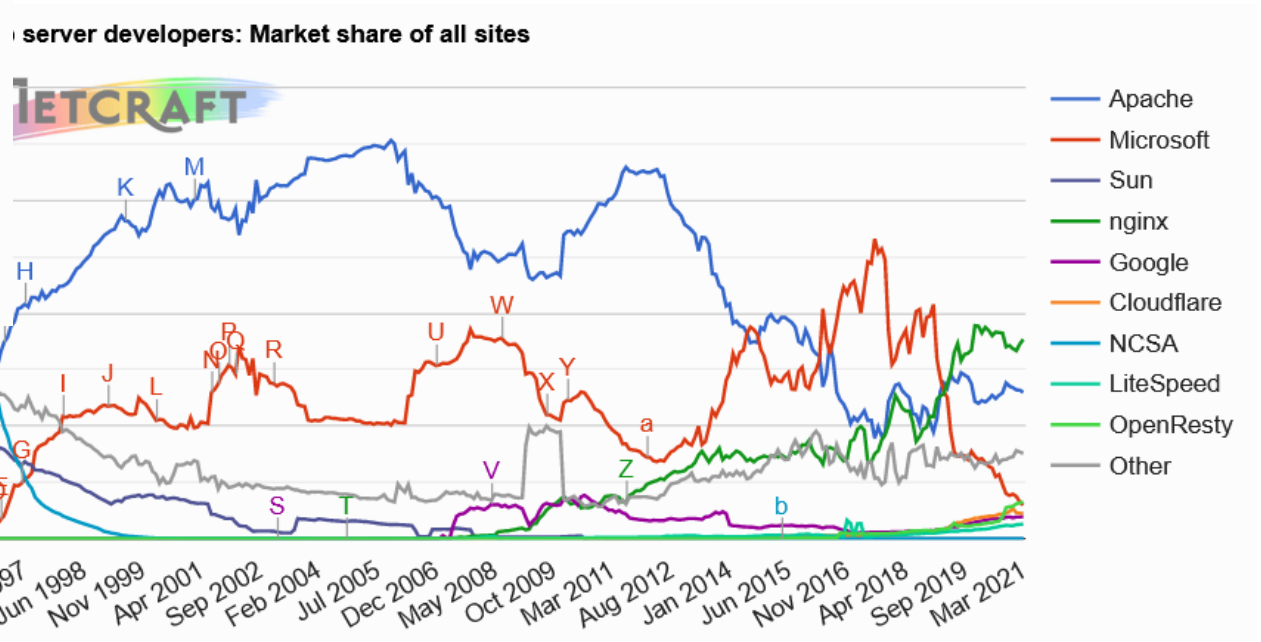
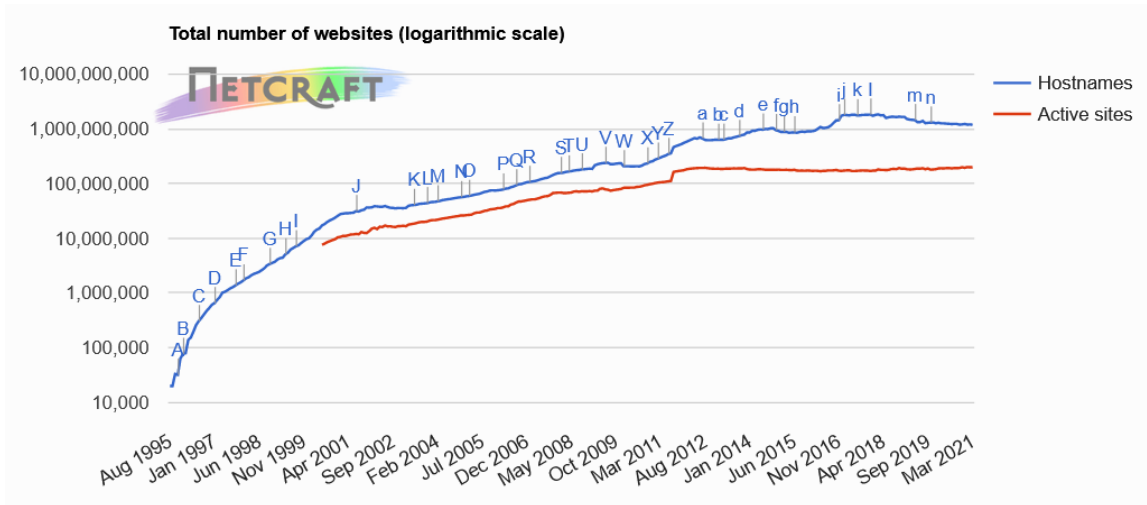
# Sessions in PHP



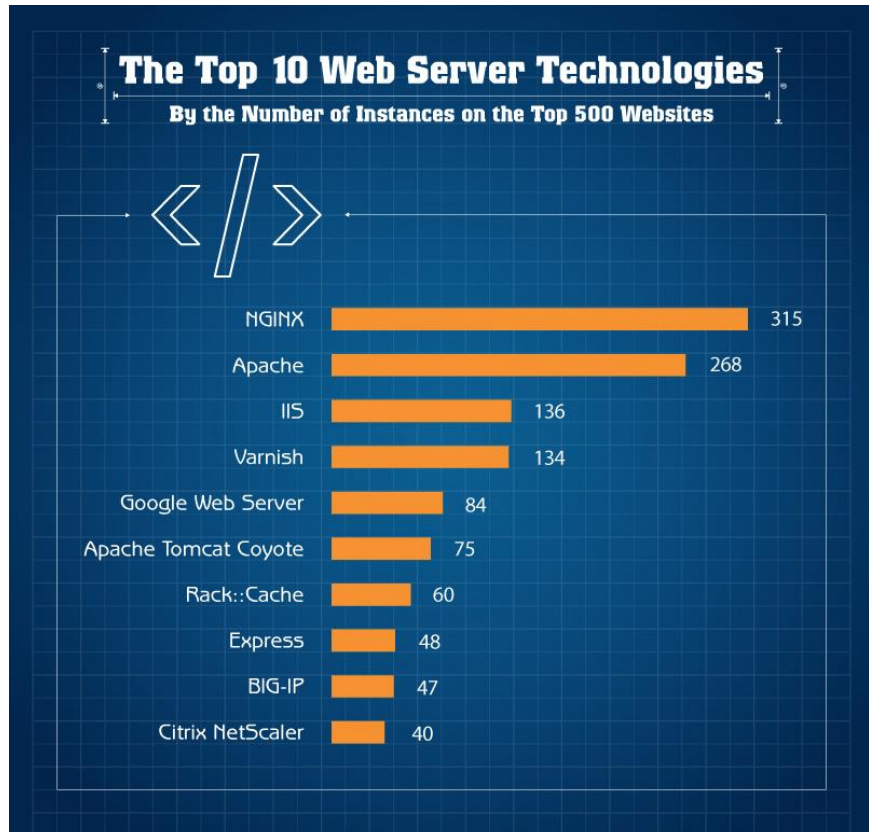


# Web Server (Allgemein)

# Websites – März 2021



# Webserver 2021



Source:  WPengine



## nginx +








- Performance! Streaming...!
- Statische Dateien!

## Apache +

- .htaccess
- Viele Funktionen!

Beide zusammen? .... ;-)

# Nginx

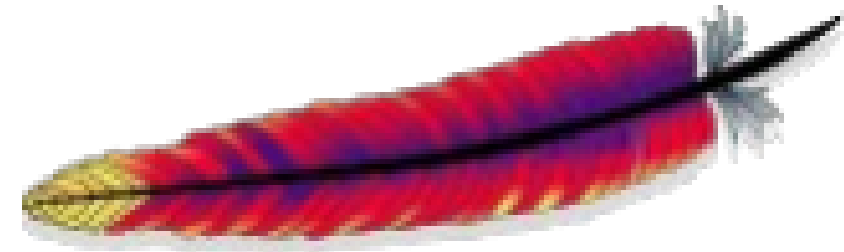
-  <http://nginx.org>
-  BSD-License
-  HTTP, E-Mail Proxy
-  Russland
-  Fast, stable, scalable
-  Module: Load Balancing,...
-  Linux, BSD, Solaris, Mac OSX, Windows XP, Server 2003

NGINX™



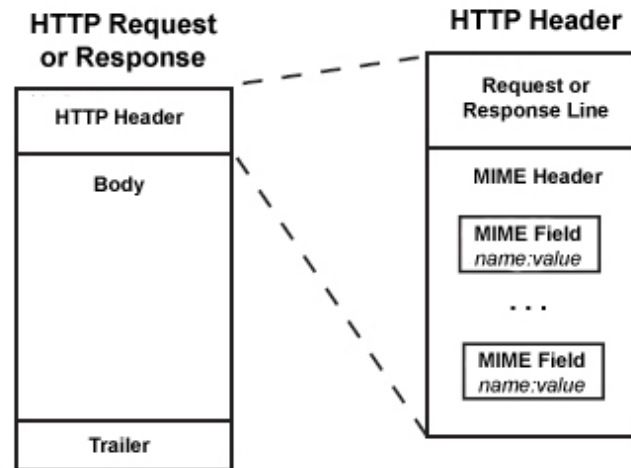
# Apache

- 🌐 <http://www.apache.org>
- 🌐 Open Source (Apache Licence)
- 🌐 Ursprünglich basierend auf NCSA HTTPd
- 🌐 Unterstützt Perl, Python, Tcl, and PHP
- 🌐 Unterstützt viele weitere Technologien als Plugins/Module
- 🌐 Load Balancing
- 🌐 Proxying



# Protokoll HTTP

- 🌐 **Stateles**
- 🌐 **Request – Response**
- 🌐 **Header – Body**
- 🌐 **Request Methoden**  
(Die wichtigen sind ...)
- **GET**
- **POST**
- 🌐 **Protokoll Versionen**
  - HTTP/1.0 (RFC1945) – Pro Anfrage eine neue TCP Verbindung
  - HTTP/1.1 (RFC2616) – Persistente Verbindung möglich
  - HTTP/2.0 (RFC7540/1) – Anfragen zusammenfassen, Kompression, Binärdaten übertragen, Push-Verfahren (Server initiiert Komm.)



Methoden:

```

Method = "OPTIONS" ;
        | "GET" ;
        | "HEAD" ;
        | "POST" ;
        | "PUT" ;
        | "DELETE" ;
        | "TRACE" ;
        | "CONNECT" ;
        | extension-method ;
extension-method = token
  
```

# HTTP Methoden



## GET

- Fordert unter Angabe einer URL eine Ressource vom Server.
- Daten werden in der URL übertragen.



## POST

- Sendet beliebige Datenmenge an Server im Inhalt (Body) einer Nachricht



## HEAD

- Verlangt nur den Kopf (Header) einer Nachricht vom Server



## PUT

- Zum Hochladen einer Ressource unter Angabe der Ziel-URL
- Damit können Ressourcen auch modifiziert werden



## DELETE

- Löscht eine Ressource am Server



## TRACE

- Lieferte einen Request zurück, wie ihn der Server bekommen hat.
- Zum Debuggen verwendet- aber Sicherheitsrisiko!



## OPTIONS

- Liefert vom Server unterstützte Methoden und Merkmale



## CONNECT

- Für HTTP Tunnel mittels Proxy- Leitet die TCP Verbindung weiter-
- ab dann nur mehr TCP und kein HTTP mehr



# Features

## HTTP Cache

- **Zwischenspeichern** von Ressourcen am Client
- Feld: "Cache-Control:" Feld
  - Kein Caching (no-store)
  - Ablaufzeit (max-age=xxxx)
  - Validierung mit Cache (must-revalidate)
  - Privat und Öffentlich (private, public)  
Einzelbenutzer vs. Proxy

## Same Origin Policy

- **Schutz vor Angriffen**- von einer Site können nur **Anfragen an die gleiche Herkunft** (*Protokoll, Domain, Port, URL*) gestellt werden
- Ausnahme sind Subdomains
- Feld "**Access-Control-Allow-Origin**":
  - \*
  - <origin>

CORS - Cross Origin Resource Sharing

`Access-Control-Allow-Origin: https://developer.mozilla.org`

# Protokollbeispiel 1

## GET Request

```
GET /test?name=Mathias HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64;
rv:45.0) Gecko/20100101 Firefox/45.0
Accept:
text/html,application/xhtml+xml,application/xml;
q=0.9,*/*;q=0.8
Accept-Language: de,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
```

Connection: keep-alive

## POST Request

```
POST /test HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64;
rv:45.0) Gecko/20100101 Firefox/45.0
Accept:
text/html,application/xhtml+xml,application/xml;
q=0.9,*/*;q=0.8
Accept-Language: de,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 12
Connection: keep-alive
```

**name=Mathias**

# Protokollbeispiel 2

## OPTIONS Request

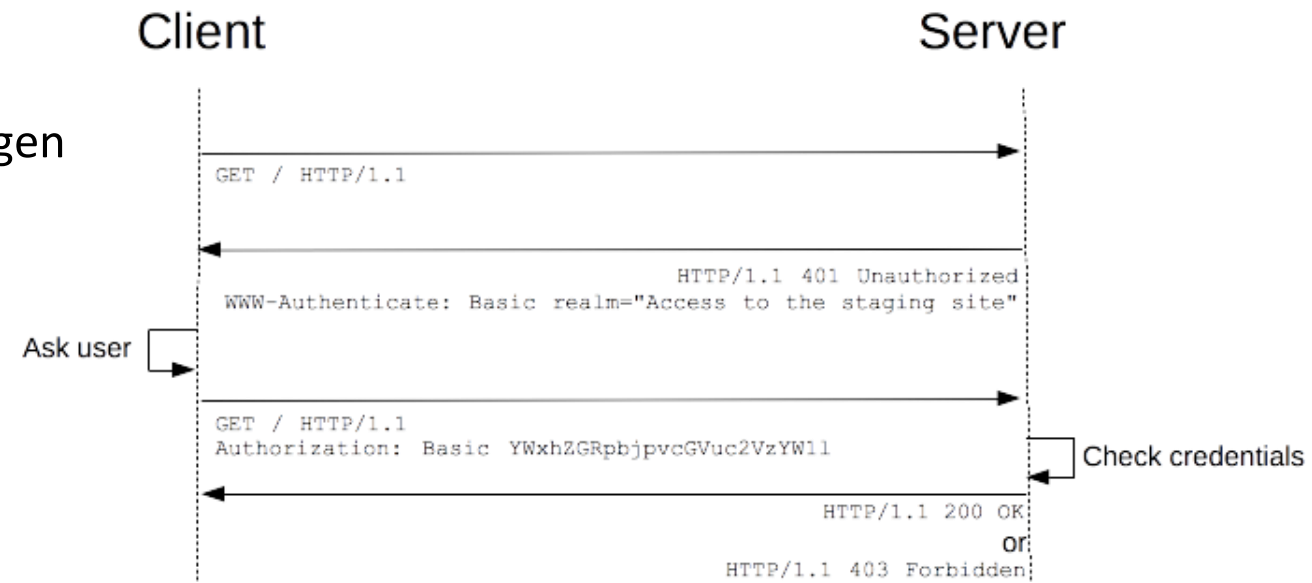
```
OPTIONS /test/ HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64;
rv:45.0) Gecko/20100101 Firefox/45.0
Accept:
text/html,application/xhtml+xml,application/xml;
q=0.9,*/*;q=0.8
Accept-Language: de,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

## OPTIONS Response

```
HTTP/1.1 200 OK
Date: Fri, 12 May 2015 11:35:43 GMT
Server: Apache/2.4.18 (Win32) OpenSSL/1.0.2e
PHP/7.0.4
Allow: GET,HEAD,POST,OPTIONS,TRACE
Content-Length: 0
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Content-Type: text/html; charset=utf-8
```

# Authentifizierung & Autorisierung 1

- 🌐 Feld "WWW-Authenticate <Typ> <Bereich>":
  - **Basic**  
Klartext – Verschlüsselung muss über HTTPS erfolgen  
Base64 kodiert
  - **Digest Access**  
Prüfsumme (Hashcode) aus Daten des Headers  
Name, Kennwort, Enthaltene Zeichenfolgen,  
Methode, usw. Mit MD5 als Standard unsicher
  - **Bearer**  
Oauth Bearer Token (IT & Mobile Security Thema)
  - **und andere!**
- 🌐 Feld "Authorization <Typ> <Name:Kennwort>":
  - Name:Kennwort ist unter Basic BASE64 kodiert



# Cookies

- 🌐 Feld "Set-Cookie <name>=<wert>":
  - Für Session Management, Personalisierung, Tracking
  - Direktiven definieren u.a. Ablaufzeit ("Expires")

- 🌐 Feld "Cookie <Liste Name Wert Zuweisungen>:

```
Set-Cookie: PHPSESSID=298zf09hf012fh2;  
Expires=Wed, 21 Oct 2019 07:28:00 GMT
```

```
Cookie: PHPSESSID=298zf09hf012fh2;
```

# Virtual Hosting (VH)




Es gibt

**Namensbasiertes**

**Portbasiertes**

**IP-basiertes**

Virtual Hosting um ...

-  ... Hosting multipler Domänen auf einem Server zu betreiben
-  ... Ressourcen effizient einzusetzen
-  ... für viele Eventualitäten Sites zu konfigurieren

# Namensbasiertes VH

- 🌐 **Unterschiedliche Domännennamen**  
verweisen auf **unterschiedliche**  
**Verzeichnisse auf einem Server**
- 🌐 **Der Port ist immer der gleiche**

```
# Ensure that Apache listens on port 80
Listen 80

# Listen for virtual host requests on all IP addresses
NameVirtualHost *:80

<VirtualHost *:80>
    DocumentRoot /www/example1
    ServerName www.example.com

# Other directives here

</VirtualHost>

<VirtualHost *:80>
    DocumentRoot /www/example2
    ServerName www.example.org

# Other directives here

</VirtualHost>
```

<http://httpd.apache.org/docs/2.2/vhosts/examples.html>

# Port-basiertes VH

- 🌐 **Unterschiedliche Ports** verweisen auf **unterschiedliche Verzeichnisse auf einem Server**, können aber den gleichen Domännennamen haben
- 🌐 **Der Port ist unterschiedlich**

*Hier im Beispiel eine Kombination aus namensbasiertem und portbasierten VH*

```
Listen 80
Listen 8080

NameVirtualHost 172.20.30.40:80
NameVirtualHost 172.20.30.40:8080

<VirtualHost 172.20.30.40:80>
  ServerName www.example.com
  DocumentRoot /www/domain-80
</VirtualHost>

<VirtualHost 172.20.30.40:8080>
  ServerName www.example.com
  DocumentRoot /www/domain-8080
</VirtualHost>

<VirtualHost 172.20.30.40:80>
  ServerName www.example.org
  DocumentRoot /www/otherdomain-80
</VirtualHost>

<VirtualHost 172.20.30.40:8080>
  ServerName www.example.org
  DocumentRoot /www/otherdomain-8080
</VirtualHost>
```

<http://httpd.apache.org/docs/2.2/vhosts/examples.html>



## IP-basiertes VH

- 🌐 **Unterschiedliche IPs auf unterschiedliche Verzeichnisse auf einem Server**, können aber den gleichen Domännennamen haben
- 🌐 Beide werden am selben Server gehostet, haben aber unterschiedliche IPs

Listen 80

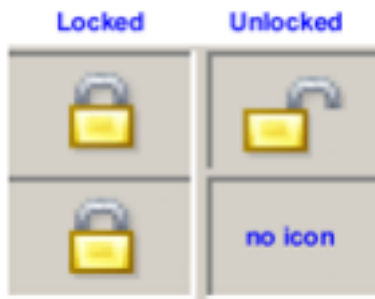
```
<VirtualHost 172.20.30.40>  
    DocumentRoot /www/example1  
    ServerName www.example.com  
</VirtualHost>
```

```
<VirtualHost 172.20.30.50>  
    DocumentRoot /www/example2  
    ServerName www.example.org  
</VirtualHost>
```

<http://httpd.apache.org/docs/2.2/vhosts/examples.html>

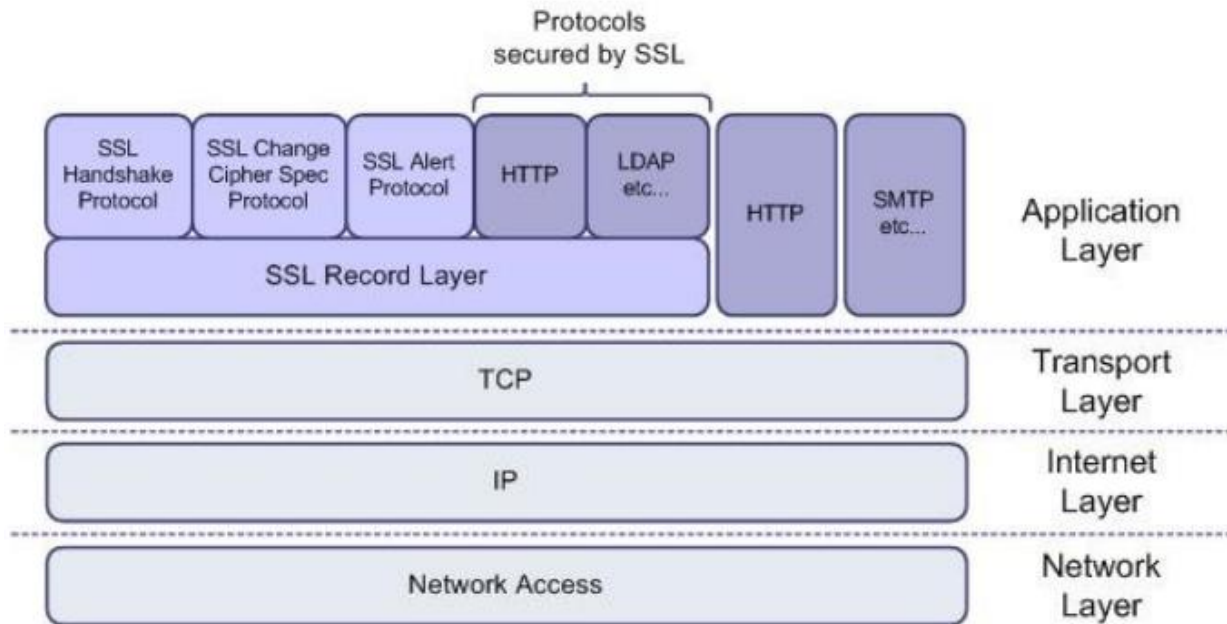
# Transport Layer Security (TLS)

- 🌐 Secure Socket Layer (alt)
- 🌐 Transport Layer Security (TLS)
- 🌐 HTTPS Port 443
- 🌐 Sichere Kommunikation
  - Authentication
  - Encryption
- 🌐 State full Connection
- 🌐 Handshake Prozedur
  - Client sendet Liste unterstützter “Cipher” Suiten
  - Server wählt die stärkste
  - Server sendet Identifikation (Digitales Zertifikat)  
*Server name, CA, public key*
  - Client kann die CA kontaktieren, um die Validität zu überprüfen
  - “Key exchange” mit “Public key cryptography”



# Transport Layer Security (TLS) | OSI & Handshake

## TLS im OSI Referenzmodell



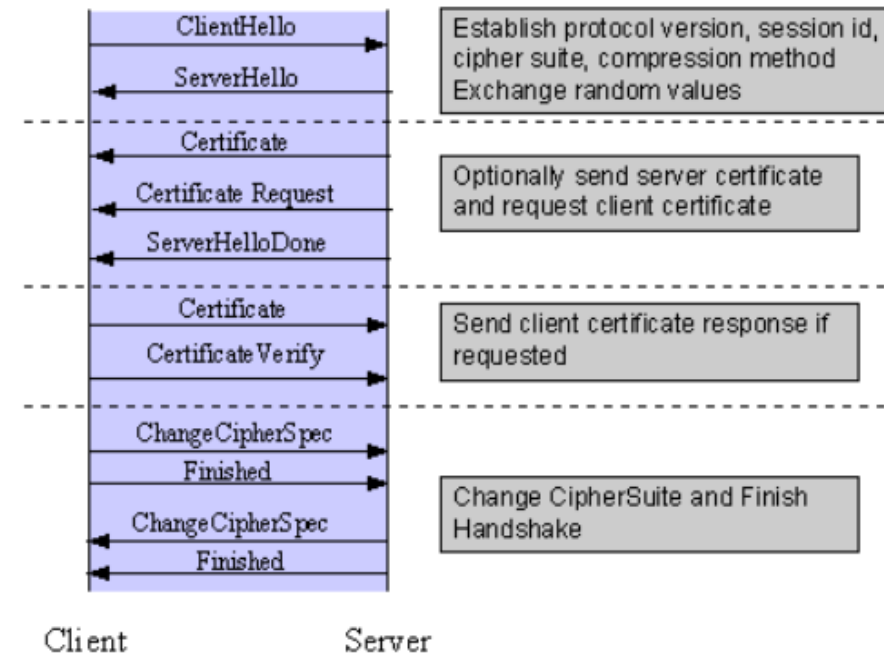
<http://www.securityfocus.com>

## TLS Handshake Protocol

Identifikation & Authentifizierung

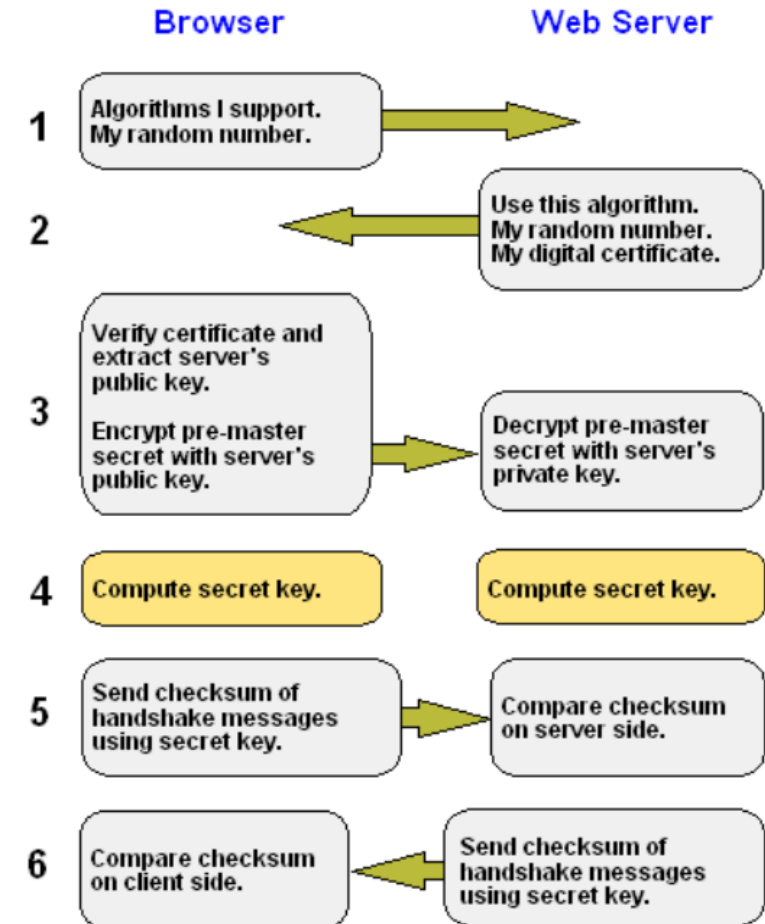
Algorithmen und Schlüssel werden ausgetauscht

4 Phasen



# Transport Layer Security (TLS) | Handshake

1. Der Client sendet ein "Hello" an den Server – zusätzlich werden die Fähigkeiten des Clients übertragen
2. Der Server antwortet mit seinen Fähigkeiten und seinem Zertifikat, das von einem Root-Zertifikat signiert wurde
3. Der Client überprüft das Zertifikat und sendet einen Vorabschlüssel, verschlüsselt mit dem öffentlichen Schlüssel des Servers
4. Beide generieren den gemeinsamen Schlüssel
5. Beide vergleichen die Nachrichten mittels Prüfsummen auf beiden Seiten (6)





**NGINX**

**Sichere Webserver**

# Open Web Application Security Project (OWASP)

- 🌐 URL: <https://owasp.org/>
- 🌐 <https://owasp.org/www-project-top-ten/>
- 🌐 Nonprofit foundation that works to improve the security of software

## Top Ten Web Application Security Risks\*

1. Injection
2. **Broken Authentication**
3. **Sensitive Data Exposure**
4. XML External Entities
5. **Broken Access Control**
6. Security Misconfiguration
7. Cross Site Scripting (XSS)
8. Insecure Deserialization
9. Using Components with known vulnerabilities
10. **Insufficient Logging & Monitoring**

Transport Layer Security  
HTTP Strict Transport Security  
(HSTS)

Authentication  
Session Management

Cross Origin Resource Sharing  
(CORS)

"A repeatable hardening process that makes it fast and easy to deploy another environment that is properly locked down. **Development, QA, and production environments should all be configured identically**, with different credentials used in each environment. **This process should be automated to minimize the effort required to setup a new secure environment.**"

[https://owasp.org/www-project-top-ten/OWASP\\_Top\\_Ten\\_2017/Top\\_10-2017\\_A6-Security\\_Misconfiguration](https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A6-Security_Misconfiguration)

# Header

## Header

- Standardmäßig geben Dienste viele Innerna preis
- Diese müssen in der Konfiguration deaktiviert werden

### Apache 2

```
HTTP/1.1 200 OK
Date: Mon, 17 Feb 2020 13:11:30 GMT
Server: Apache/2.4.38 (Debian)
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 24275
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

*Date: /etc/apache2/conf-available/security.conf*

```
ServerTokens Prod
ServerSignature Off
```

### Nginx

```
HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Tue, 12 Nov 2019 13:04:05 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
Content-Encoding: gzip
```

*Datei: /etc/nginx/nginx.conf*

```
server_tokens off;
```

### Node

```
HTTP/1.1 200 OK
X-Powered-By: Express
x-timestamp: 1581943631836
Accept-Ranges: bytes
Cache-Control: public, max-age=86400
Last-Modified: Mon, 17 Feb 2020 10:59:04 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 188
Date: Tue, 12 Nov 2019 13:01:03 GMT
Connection: keep-alive
```

*Datei: /srv/node-server/index.js*

```
setHeaders: function (res, path, stat) {
  res.set('x-timestamp', Date.now());
  res.removeHeader("X-Powered-By");
}
```

# Directory

 **Verzeichnisinhalte** werden bei manchen Servern automatisch angezeigt

 **Server**

- **Apache**

- <Directory /var/www/html/ordner>

- Options Indexes**




- oder

- Options None**

- **Nginx**

- Standardmäßig deaktiviert!

## Index of /ordner

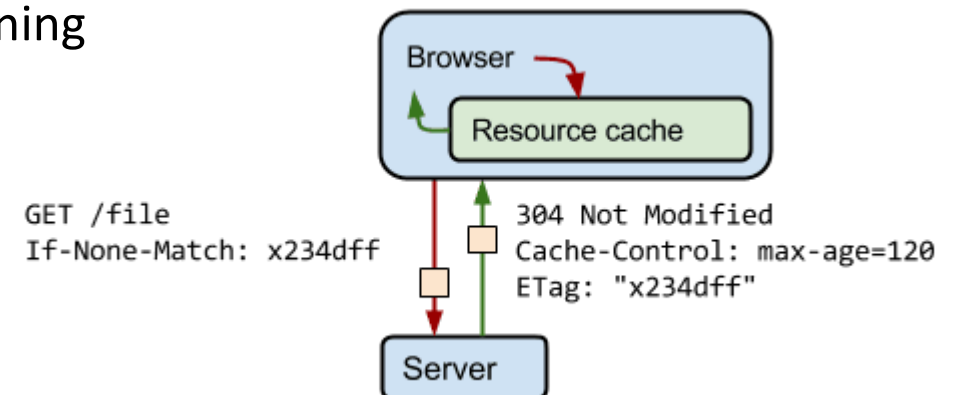
<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 <a href="#">Parent Directory</a>		-	
 <a href="#">benutzerdaten.txt</a>	2021-02-02 10:28	0	
 <a href="#">secret.docx</a>	2021-02-02 10:28	0	

*Apache/2.4.38 (Debian) Server at 10.52.200.95 Port 80*



# ETags

- 🌐 In HTTP Response- um einer **Ressource über eine spezifische Version zu identifizieren**
- 🌐 Ist **üblicherweise ein Hashwert aber in alten Versionen von Apache kann man daraus jedoch Systeminformationen** (iNode Nummer, etc) extrahieren
- 🌐 Heute werden ETags u.a. für "**Cookieless Cookies**" verwendet- User Tracking:
  - <https://github.com/lucb1e/cookielesscookies>
  - [https://owasp.org/www-community/attacks/Cache\\_Poisoning](https://owasp.org/www-community/attacks/Cache_Poisoning)
- 🌐 **Server**
  - Apache: **FileETag None**
  - Nginx: **etag off**
  - Node: **app.set('etag', false);**



# Einstellungen auf Verzeichnisebene (Apache)

🌐 In **Ordern** können standardmäßig Konfigurationen überschrieben werden:  
Versteckte Datei **".htaccess"**

🌐 Überschreiben verhindern:

```
<Directory /var/www/>
```

```
    AllowOverride None
```

oder

```
    AllowOverride AuthConfig Indexes
```

um spezifische Konfiguration zu erlauben

# HTTP Methoden

- 🌐 Einige **HTTP Methoden bergen potentielle Sicherheitsrisiken** (z.B. "Trace")
- 🌐 HTTP Methoden können **explizit erlaubt** werden
- 🌐 Server

- **Apache2**

```
<Directory /var/www/>  
    <LimitExcept GET POST HEAD>  
        deny from all  
    </LimitExcept>
```

- **Nginx** (Anmerkung: Direktive "limit\_except" funktioniert nur mit einer Methode)

```
server {  
    if ($request_method !~ ^(GET|HEAD|POST)$ )  
    {  
        return 405; // Status: Not allowed  
    }  
}
```

# Weiterleitung zu HTTPS

## Apache

```
<VirtualHost *:80>
    ServerName alice.secwebapps.de
    DocumentRoot /var/www/html
    Redirect permanent / https://alice.secwebapps.de
</VirtualHost>
```

## Nginx

```
server {
    listen 80;
    server_name alice.secwebapps.de;
    root /var/www/html;
    return 301 https://alice.secwebapps.de;
}
```

## Node

```
app.use(function(request, response){
    if(!request.secure){
        response.redirect("https://alice.secwebapps.de");
    }
});
```

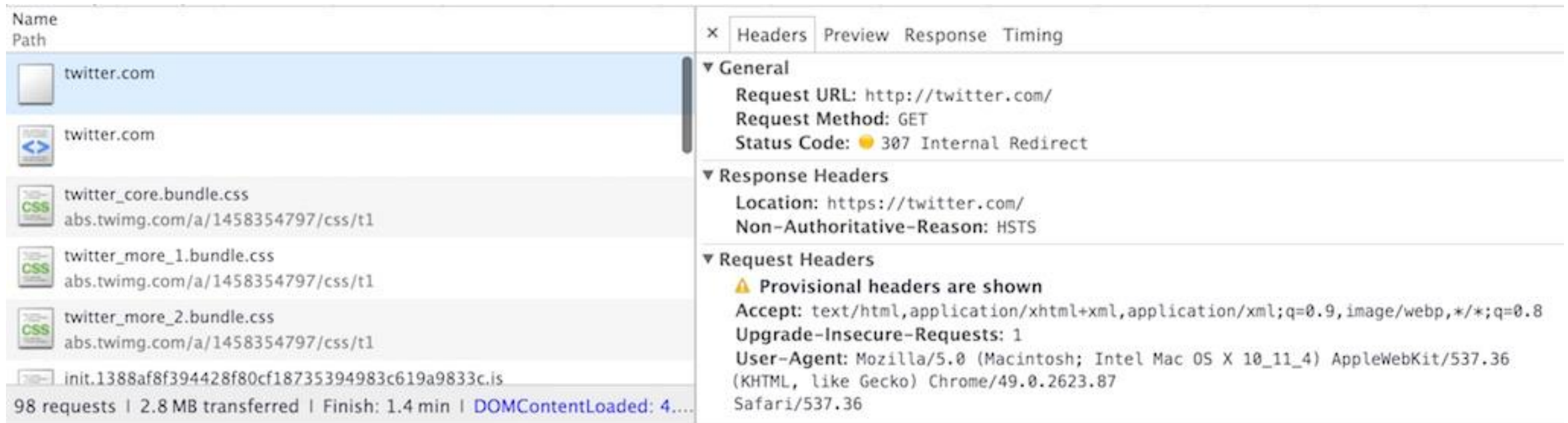
# HTTP Strict Transport Security (HSTS) 1/2






- 🌐 Web Security Policy um gegen Man-in-the-Middle Attacken zu schützen:  
Header: `strict-transport-security: max-age=31536000`
  - HTTPS zwingend erforderlich d.h. http → https
  - Bei nicht vertrauenswürdigen Zertifikaten wird die Verbindung vom Browser geschlossen

## 🌐 Servers

- Apache:  
`Header always set Strict-Transport-Security "max-age=31536000; includeSubDomains"`
- Nginx:  
`add_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always;`
- Node: (über das Modul "helmet")  
`var helmet = require('helmet');  
app.use(helmet());`

# HTTP Strict Transport Security (HSTS) 2/2



Name Path	Headers Preview Response Timing
 twitter.com	<b>General</b> Request URL: http://twitter.com/ Request Method: GET Status Code: 307 Internal Redirect
 twitter_core.bundle.css abs.twimg.com/a/1458354797/css/t1	<b>Response Headers</b> Location: https://twitter.com/ Non-Authoritative-Reason: HSTS
 twitter_more_1.bundle.css abs.twimg.com/a/1458354797/css/t1	<b>Request Headers</b> ⚠ Provisional headers are shown Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8 Upgrade-Insecure-Requests: 1 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/49.0.2623.87 Safari/537.36
 twitter_more_2.bundle.css abs.twimg.com/a/1458354797/css/t1	
 init.1388af8f394428f80cf18735394983c619a9833c.js	

98 requests | 2.8 MB transferred | Finish: 1.4 min | DOMContentLoaded: 4....

Wieso **besser als Redirect (301)**?

- Betrifft die ganze Domain
- Eigener Cache mit eigenem Timeout
- Browser umgehen HTTP damit (HTTPS-only)

# Authentifizierung

## 🌐 Mechanismen

- **Basic** | base64-encoded credentials
- **Bearer** | See RFC 6750, bearer tokens to access OAuth 2.0-protected resources
- **Digest** | MD5 hashing - unsicher
- **HOBA** | See RFC 7486, Section 3, HTTP Origin-Bound Authentication, digital-signature-based
- etc.

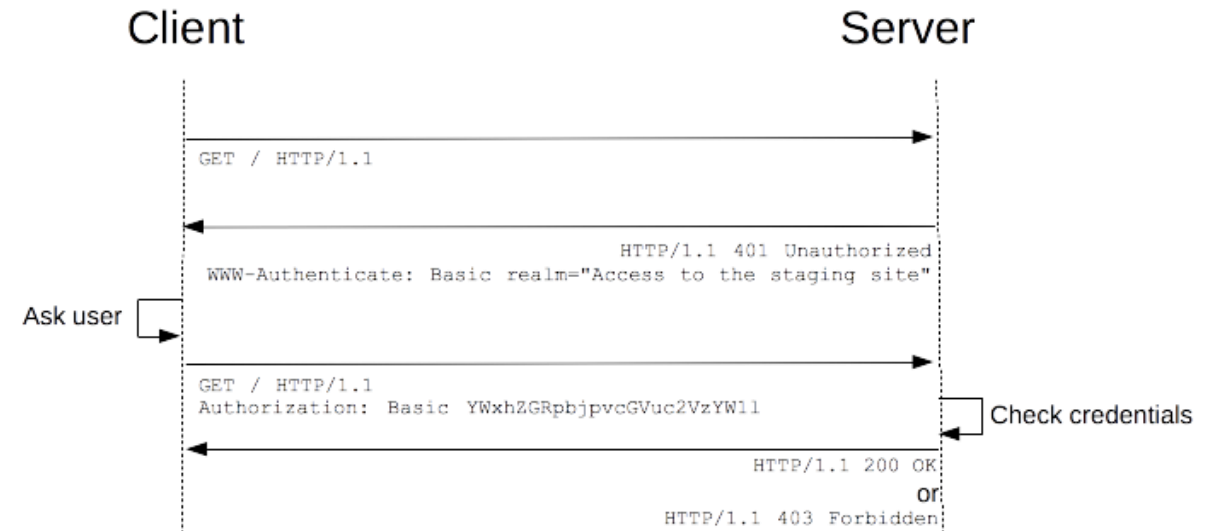
## 🌐 Apache

`AuthType Basic`

`AuthName "Access to the staging site"`

`AuthUserFile /path/to/.htpasswd`

`Require valid-user`



<https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication>

# Cookie Parameter



## HttpOnly

Das Cookie kann nicht über JavaScript ausgelesen werden



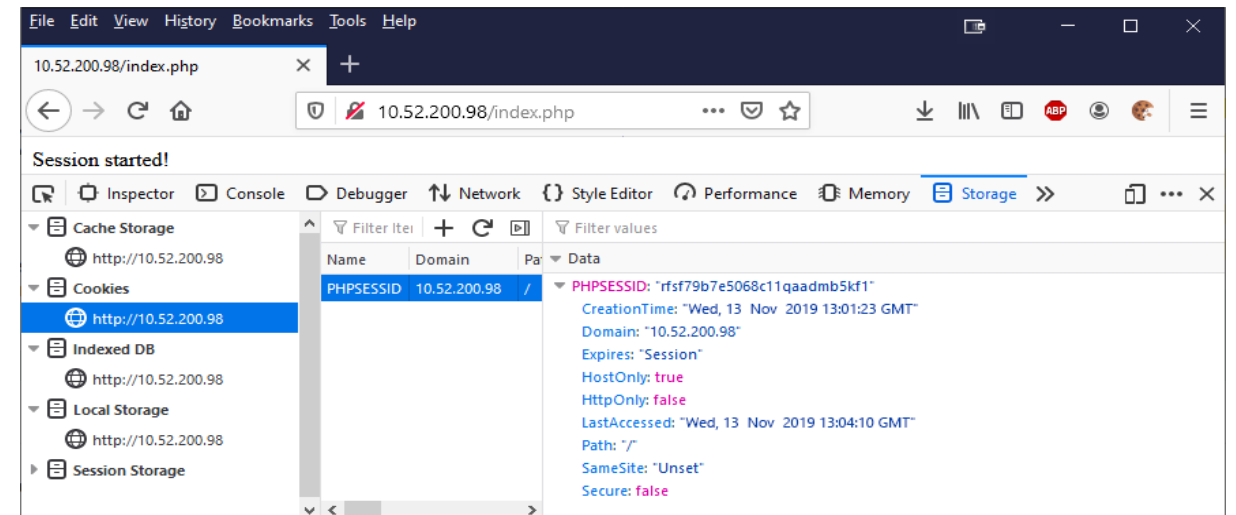
## Secure

Das Cookie wird nicht über eine ungesicherte Leitung gesendet



## SameSite

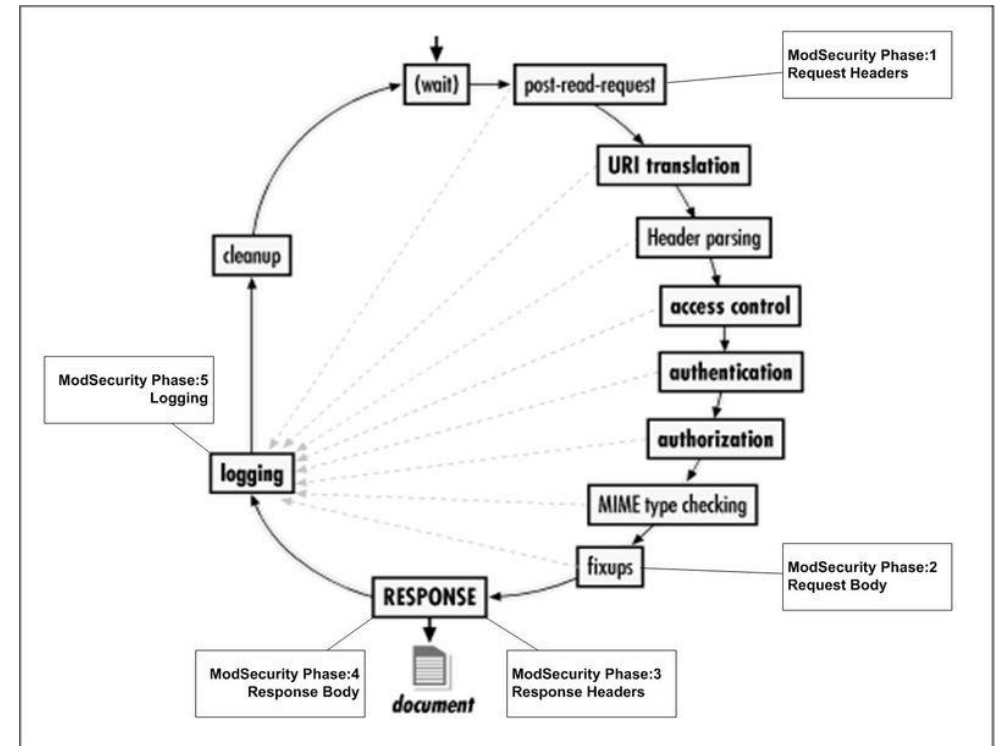
Das Cookie kann nur von der exakt selben Domain / IP abgerufen werden





# Web Application Firewall (WAF)

- 🌐 **Inspiziert die Inhalte** von Web-Requests
- 🌐 **Blockiert Anfragen** und/oder **loggt sie** mit
- 🌐 **Regeln** können auf
  - Request headers (REQUEST\_HEADERS)
  - Request body (REQUEST\_BODY)
  - Response headers (RESPONSE\_HEADERS)
  - Response body (RESPONSE\_BODY)
  - Logging (LOGGING)
 angewendet werden
- 🌐 **ModSecurity**
  - <https://modsecurity.org>
  - Muss als separates Modul auf Apache / Nginx installiert und aktiviert werden



# WAF | Aktionen



## **Disruptive**

ModSecurity muss reagieren. Z.B.: eine Transaktion blocken. Es kann nur eine disruptive Aktion pro Regel (Chain) angewendet werden. Die Aktion wird NICHT ausgeführt, wenn "SecRuleEngine" ist auf "DetectionOnly" gesetzt.



## **Non-disruptive**

Eine Aktion vornehmen ohne den Fluss der Regelbearbeitung zu unterbrechen. Z.B.: Eine Variable oder deren Wert setzen. Diese Aktionen können in jeder Regel durchgeführt werden (Chain)



## **Flow**

Beeinflusst den Weg, in dem Regeln abgearbeitet werden (Z.B.: Skip oder SkipAfter).



## **Meta-data**

Mehr Information zu einer bestimmten Regel liefern. Z.B.: id, rev, severity und msg.

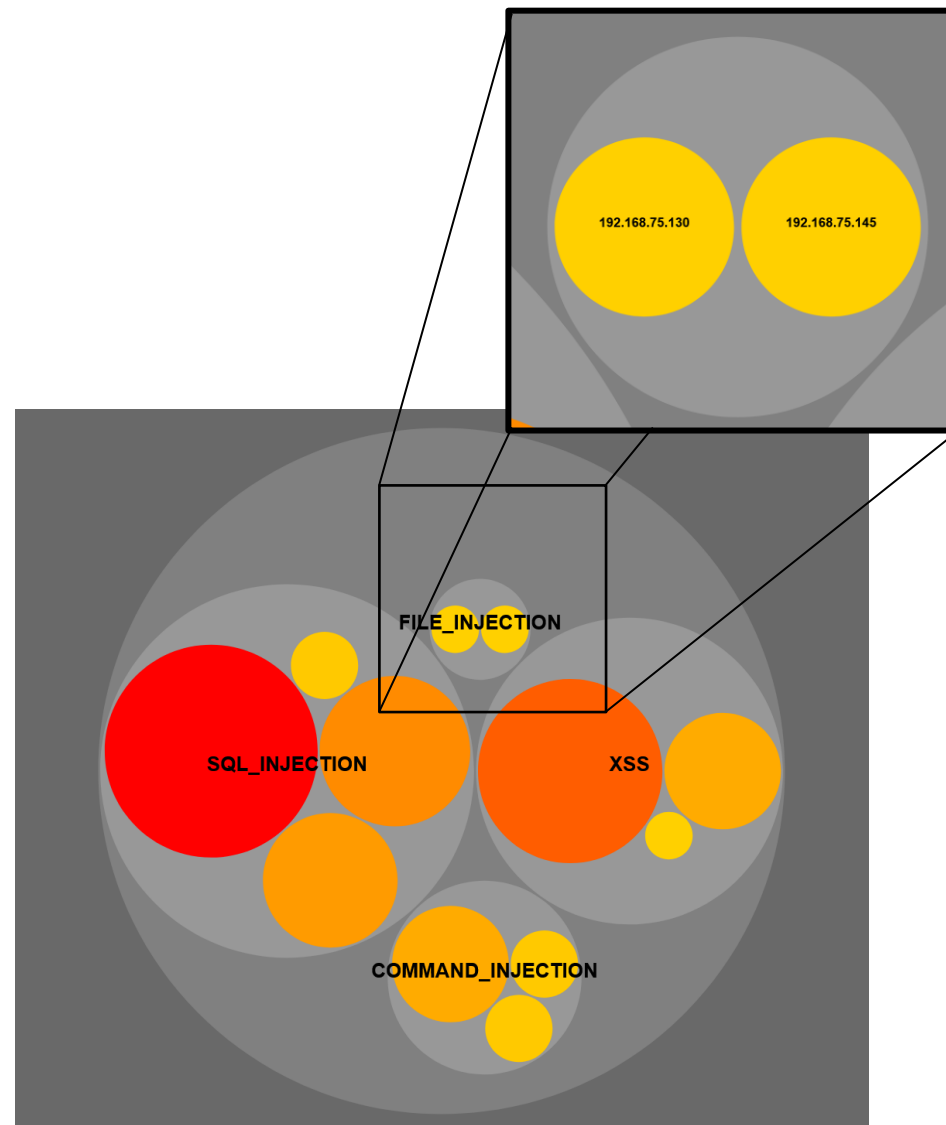
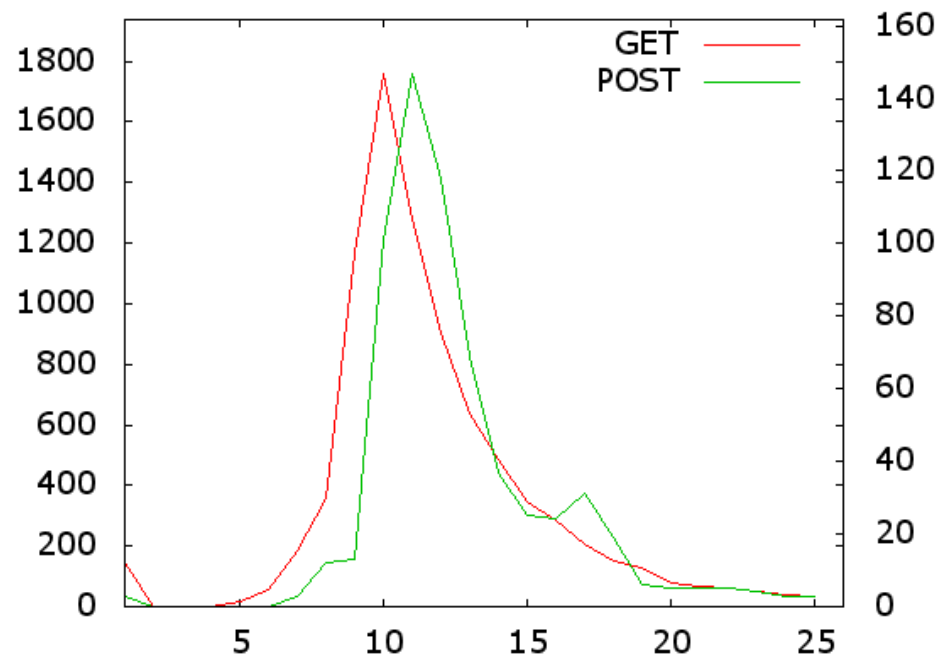


## **Data actions**

Container, die Daten für andere Aktionen beinhalten. Z.B.: Die Status-Aktion, die den Status für das Blocken beinhaltet (sofern durchgeführt).

# Visualisierungen

Duration of Requests:  
GET vs. POST





# OAuth2.0 (RFC 6749 )

in Englisch

# Open Authorization

- 🌐 **Framework for Authorization**
- 🌐 2012 endorsed successor of the 2006 developed OAuth (RFC 5849) protocol
- 🌐 Enables web applications for a limited access to HTTP services, either via ...
  - an an Authorization between Resource-Owner and Service or
  - an own, dedicated grant for a web application
- 🌐 Uses TLS/SSL (obligatory)
- 🌐 Used with mobile devices, browsers and JS applications (Rich Internet Applications)

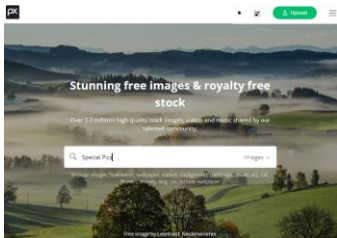
# Starting Position



User



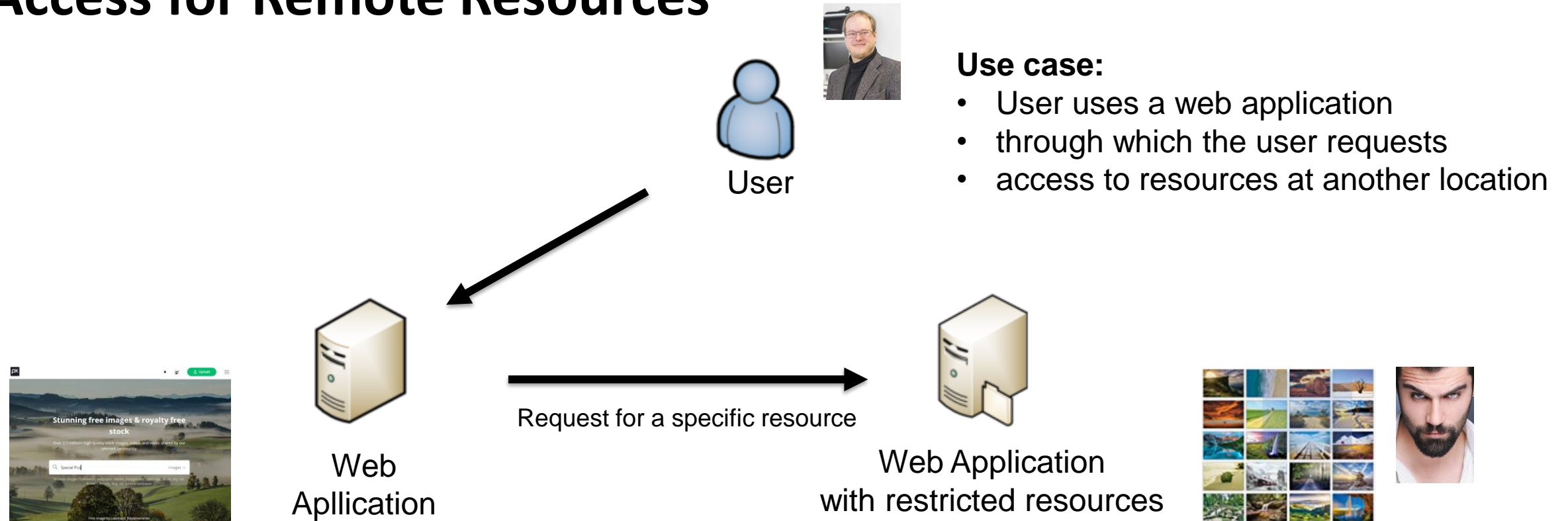
Web  
Application



Web Application  
with restricted Resources

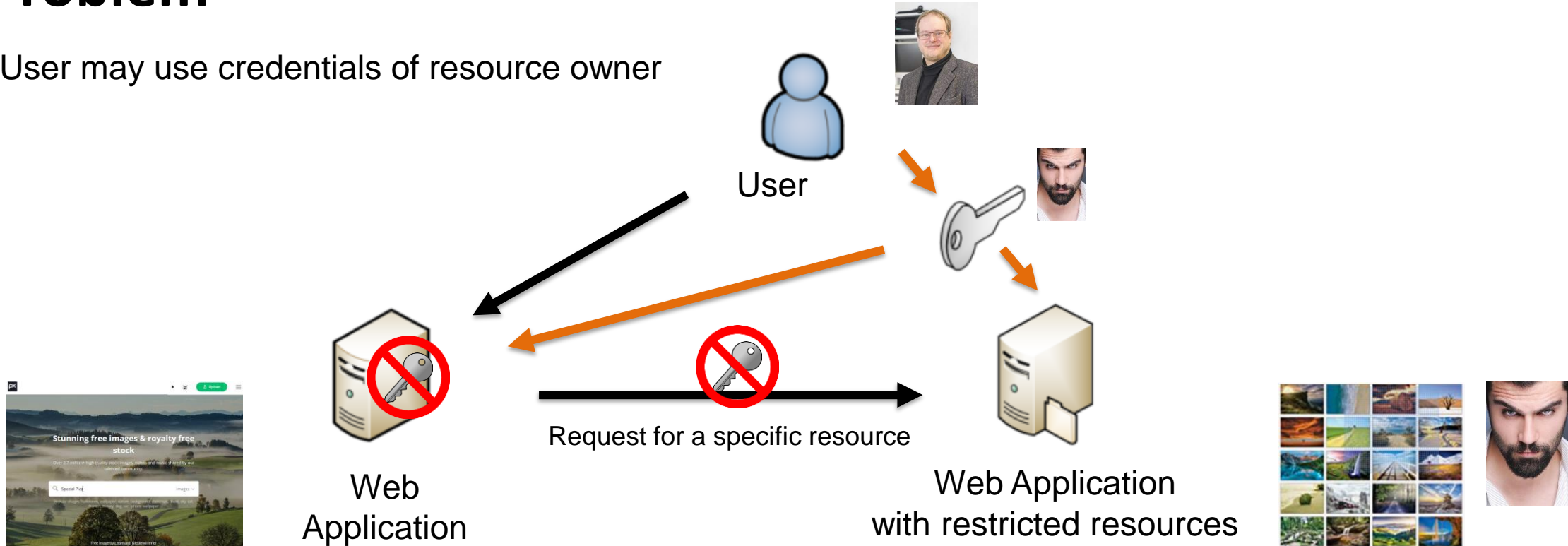


# Access for Remote Resources



# Problem

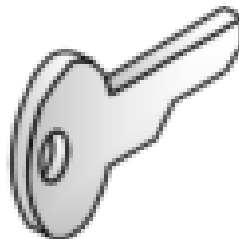
User may use credentials of resource owner






# Traditional Authentication


- 🌐 If a Client wants to access a resource, he may use the credentials of the resource owner.
- 🌐 But the disadvantages are ...
  - that the credentials of the owner have to be saved on the web application which provides resources
  - that authentication happens with keys
  - that full access to the resources of the owner is possible
  - that access once granted is hard to revoke later
  - that if the web application providing the resources is compromised, the owners credentials are compromised as well




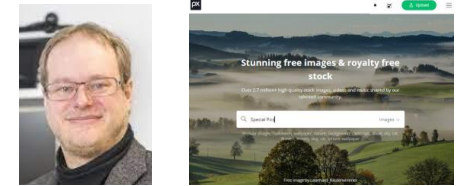
# Roles

 **Client**  
Application, which wants to access a restricted resource. Needs the **Authorization of the owner of resources**

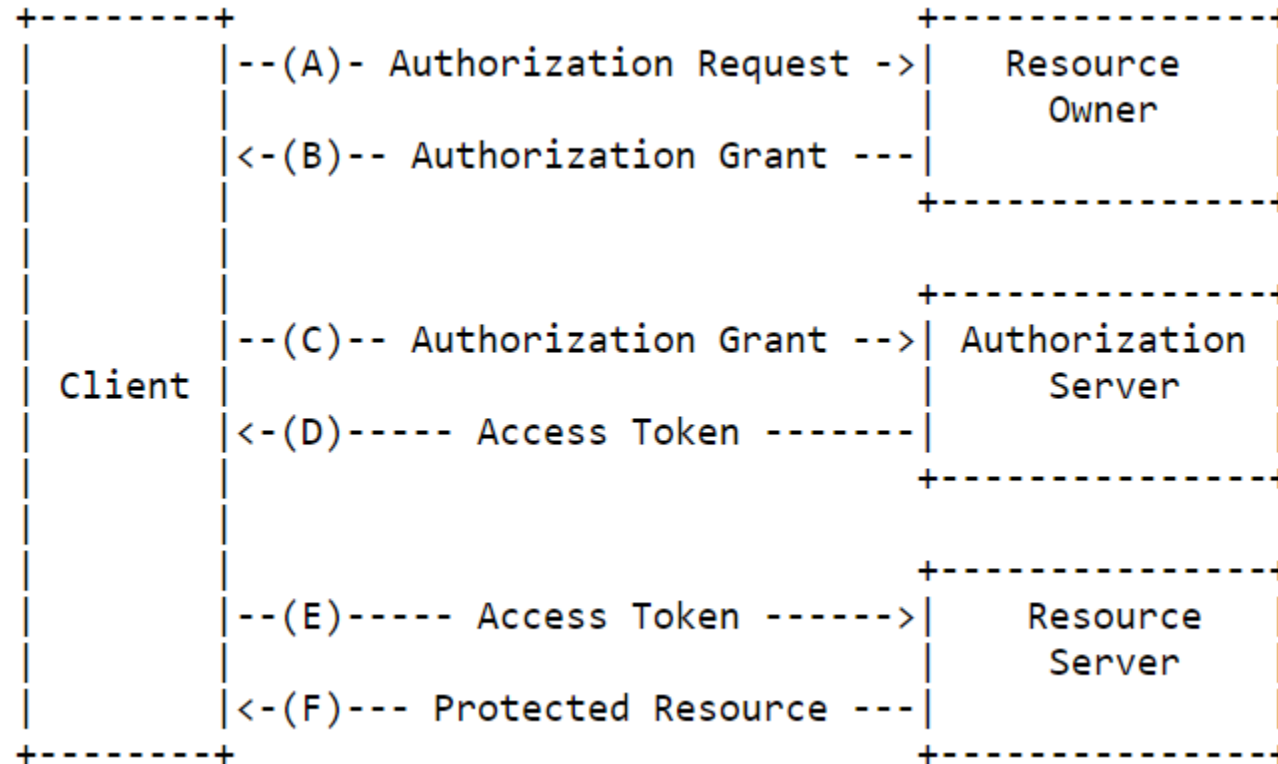
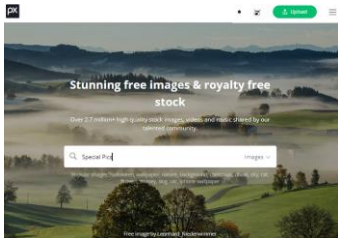
 **Resource Owner**  
A **System or a person** owning the resources

 **Authorization Server**  
Responsible for **Access-Tokens**, with which a client authenticates against the resource owner and which holds the resource token(s)

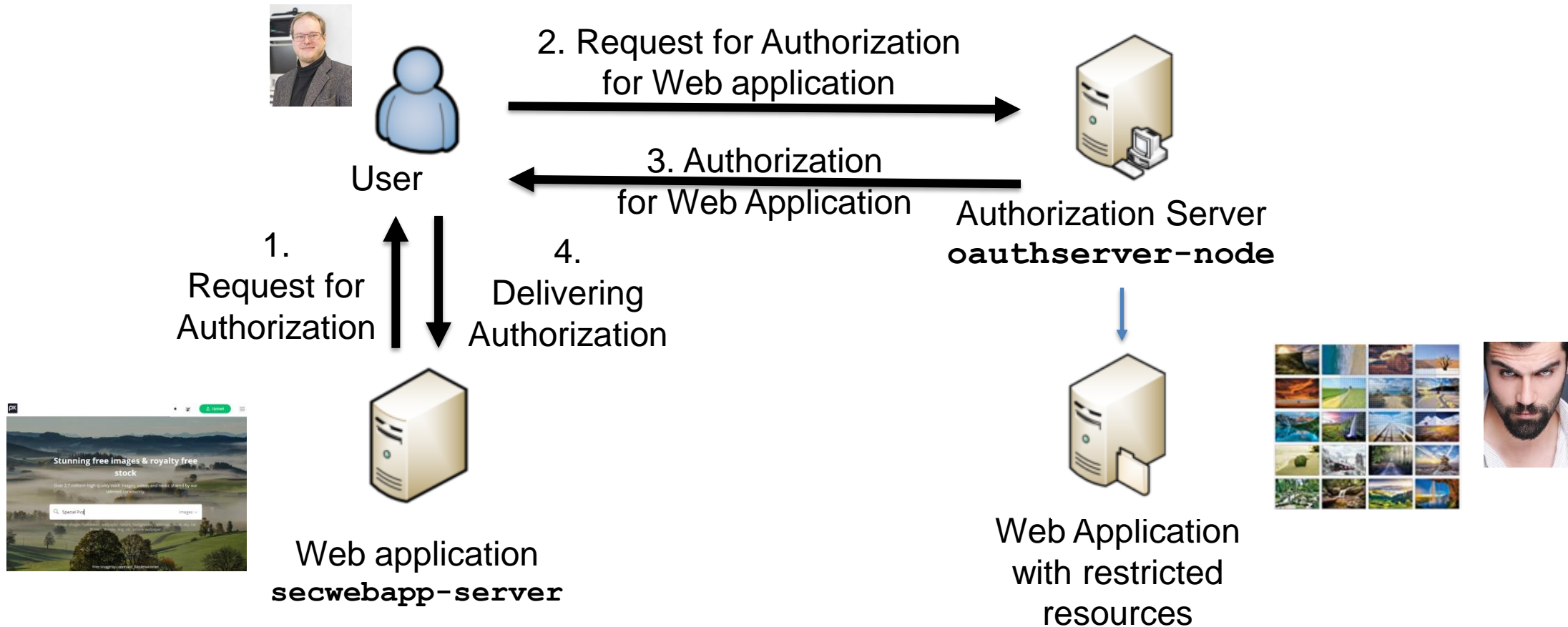
 **Resource Server**  
This role **points** to a system or a person owning the resources



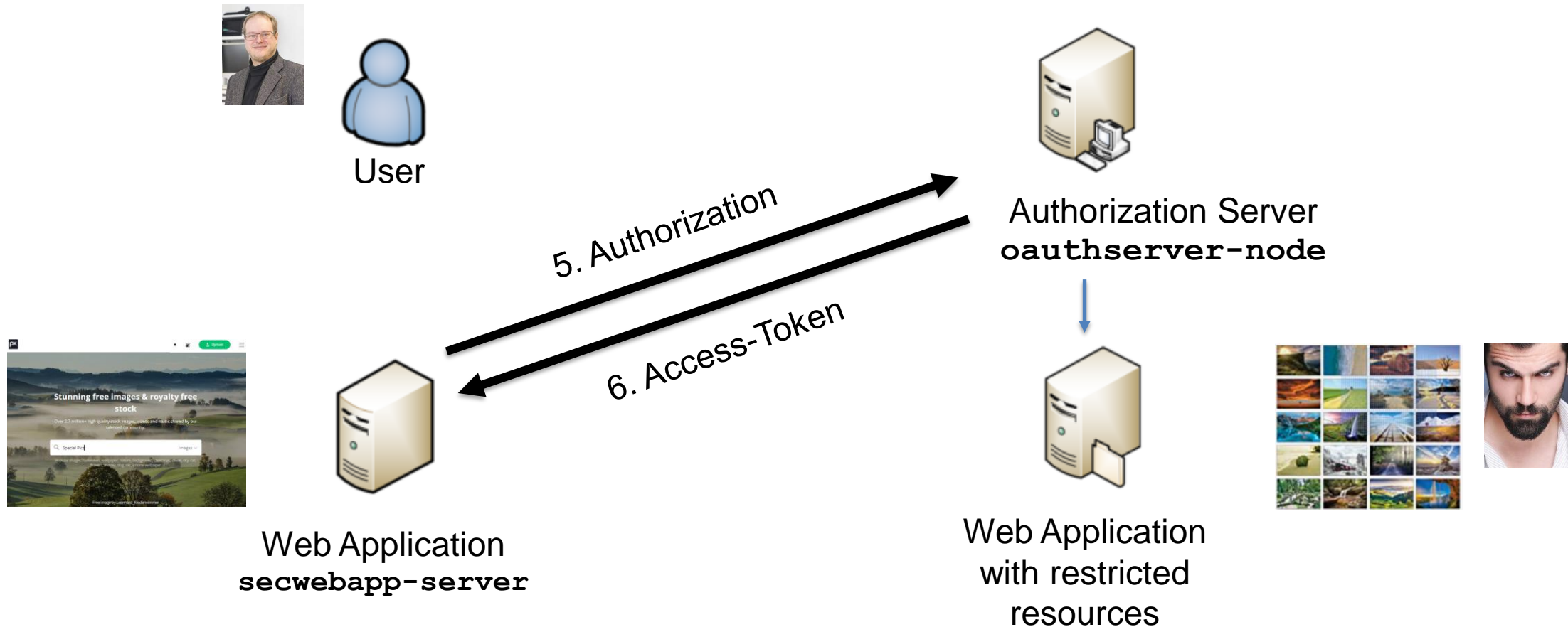
# Oauth 2 Method



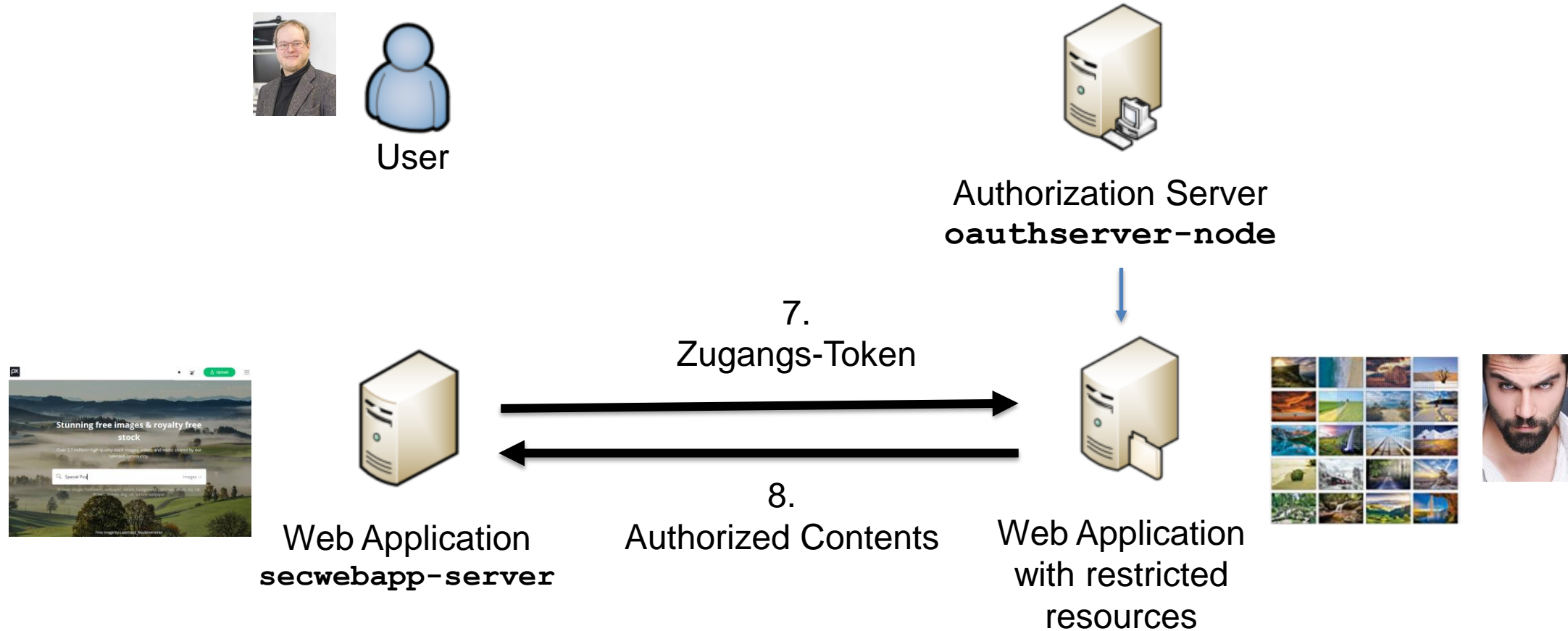
# Solution (Step 1)



## Solution (Step 2)



# Solution (Step 3)





# Register

- 🌐 The Client has to register on the Authorization Server
- 🌐 The following arguments are expected:
  - Client Type
  - Redirection URI(s)
  - API specific information
- 🌐 After successful registration, the client obtains:
  - ID
  - Secret ("Key")

# Profile

There are several Client Types:

-  Server-side Web Application
  - Key or Access-Tokens do not reside on client-side
  - The Client needs multiple access to restricted resources
-  Application with a "User Agent" and native Applications
  - Single Page Applications, Browser-Extensions, etc.
  - Handy-App
  - Protocol- and credentials are easily accessible



# Workflows

- 🌐 Server-side Web Application  
(Authorization Code)
- 🌐 User Agent (Browser) Application  
(Implicit)
- 🌐 Password of Resource Owner  
(Resource Owner Password Certificate)
- 🌐 Access Data of Client  
(Client Credentials)

# Serverside Web Application: *Authorisation Grant*

- 🌐 Two Steps:
  - Request for Authorization
  - Request for Access-Token
- 🌐 Support of "Refresh-Tokens"
- 🌐 The Authorization is used for "Access-" and "Refresh-Tokens" alike
- 🌐 Used with secure clients

# Example: OAuth 2.0 Server mit node.js

Example Implementation:

## NodeJS OAuth 2.0 Provider

*Source:* [https://github.com/Meeks91/nodeJS\\_OAuth2Example](https://github.com/Meeks91/nodeJS_OAuth2Example)

*Tutorial:* <https://blog.cloudboost.io/how-to-make-an-oauth-2-server-with-node-js-a6db02dc2ce7>

**Install / Configure** the example **on your secure web server environment!**

Use the implementation in memory at first:

```
node_modules/node-oauth2-server/examples/memory
```

## Notre:

Database implementation under

```
node_modules/node-oauth2-server/examples/
```

MariaDB Login:

```
update user set authentication_string=password('toor'), plugin='mysql_native_password' where user='root';
```

# Access-Token for Authentication

POST http://10.52.200.106:9980/auth/login Send Save

Params Authorization Headers (1) **Body** Pre-request Script Tests Cookies Code Comments (0)

none  form-data  x-www-form-urlencoded  raw  binary

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	username	knolm			
<input checked="" type="checkbox"/>	password	keines			
<input checked="" type="checkbox"/>	grant_type	password			
<input checked="" type="checkbox"/>	client_id	makno			
<input checked="" type="checkbox"/>	client_secret	keines			
	Key	Value	Description		

Body Cookies Headers (9) Test Results Status: 200 OK Time: 12 ms Size: 387 B Save Download

Pretty Raw Preview JSON ≡ 📄 🔍

```
1 {
2   "token_type": "bearer",
3   "access_token": "fbf08f2e07f159b934c6cba782ea46490ab77b22",
4   "expires_in": 3600
5 }
```

# Test of Access-Tokens

POST ▼ http://10.52.200.106:9980/restrictedArea/enter Send Save ▼

Params Authorization **Headers (2)** Body Pre-request Script Tests Cookies Code Comments (0)





	KEY	VALUE	DESCRIPTION	...	Bulk Edit	Presets
<input checked="" type="checkbox"/>	Content-Type	application/x-www-form-urlencoded				
<input checked="" type="checkbox"/>	Authorization	Bearer fbf08f2e07f159b934c6cba782ea46490ab77b22				
	Key	Value	Description			

Body **Headers (6)** Test Results Status: 200 OK Time: 4 ms Size: 239 B Save Download

Pretty Raw Preview HTML ▼ ☰ 🔍

```
i 1 You have gained access to the area
```

## URLS

-  <https://oauth.net/2/>  
Overview and implementation
-  <https://aaronparecki.com/oauth-2-simplified/#single-page-apps>  
Basic overview on OAuth 2.0
-  <https://medium.com/google-cloud/understanding-oauth2-and-building-a-basic-authorization-server-of-your-own-a-beginners-guide-cf7451a16f66>  
Another overview in workflow
-  <https://auth0.com/blog/on-the-nature-of-oauth2-scopes/>



# WebRTC

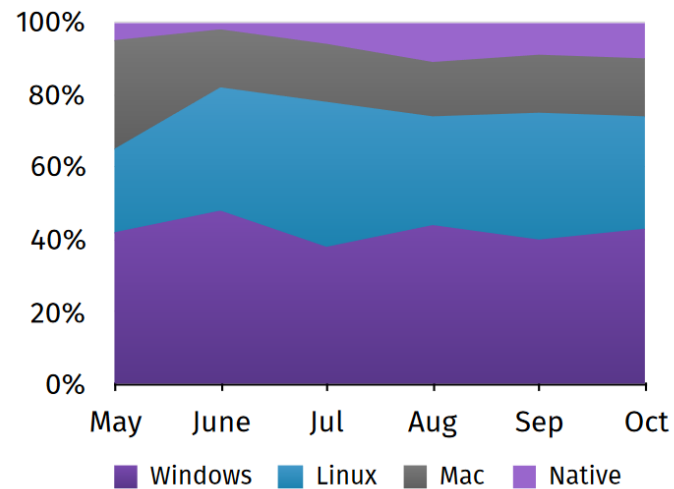
in Englisch

# Overview

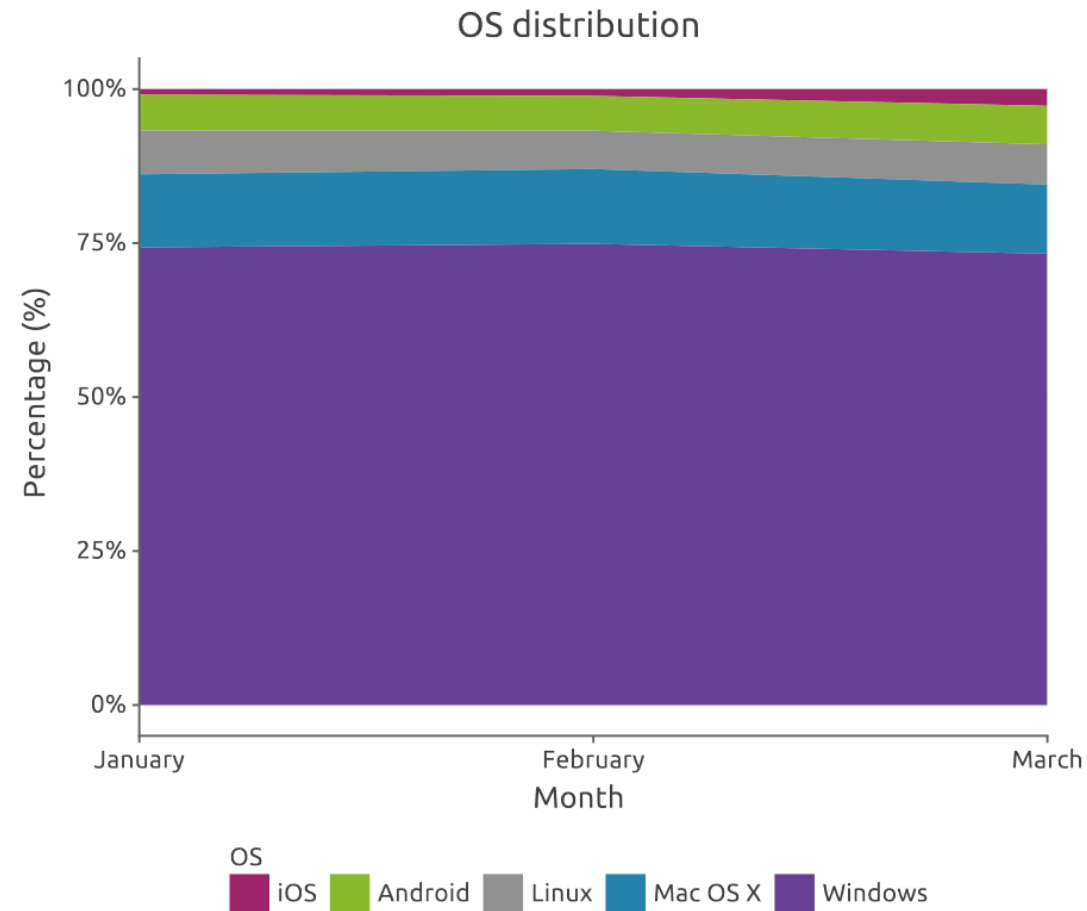
- 🌐 Web Real-Time Communication
- 🌐 2011 presented HTML5 Spezifikation by Google, Mozilla, Opera, and others
- 🌐 Most modern browsers support WebRTC (Chrome, Firefox, Safari, Edge)
- 🌐 Browser and (mobile) devices are able to exchange Data, Audio, Video over a direct Peer-to-Peer connection
- 🌐 No Plug-Ins or native applicationes are used
- 🌐 Allthough not 100% Peer-to-Peer (TURN)



# WebRTC Statistics: OS

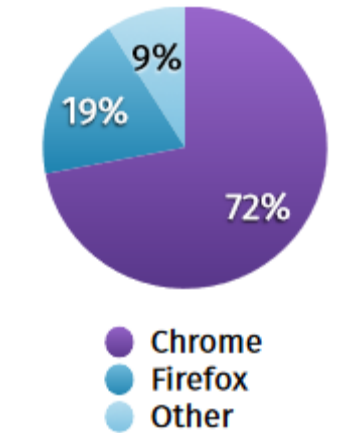


callstats.io/2016

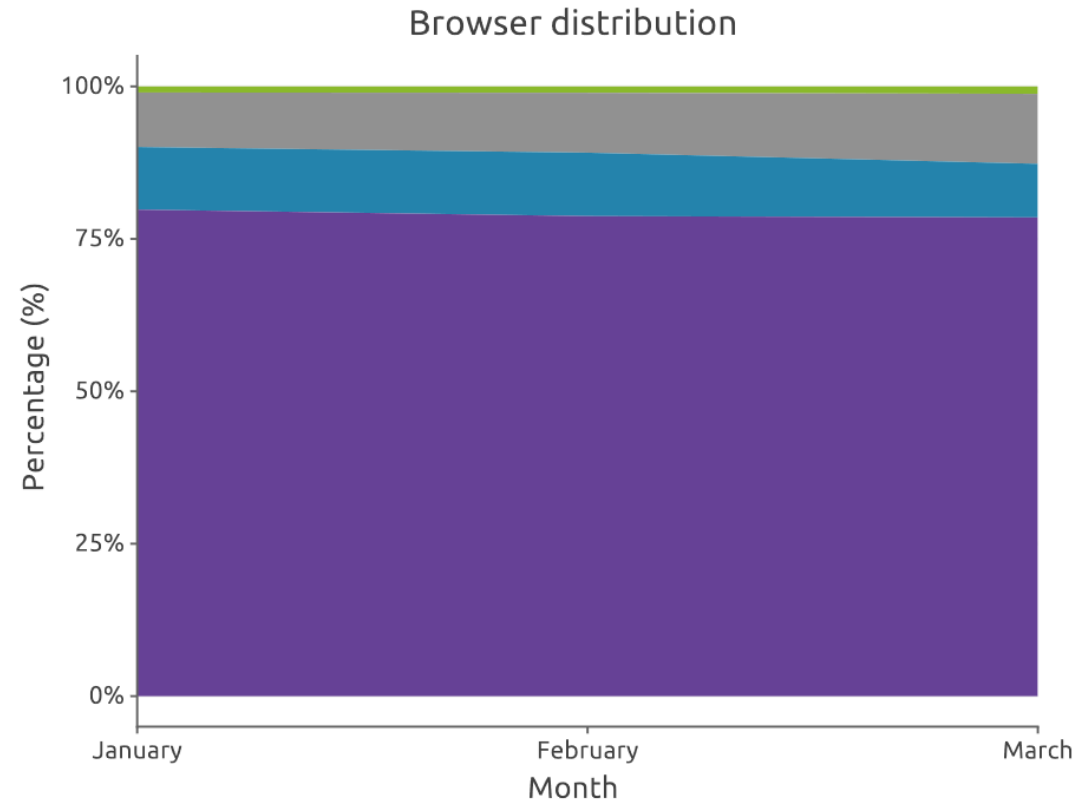


callstats.io/2018

# WebRTC Statistics: Browser



callstats.io/2016






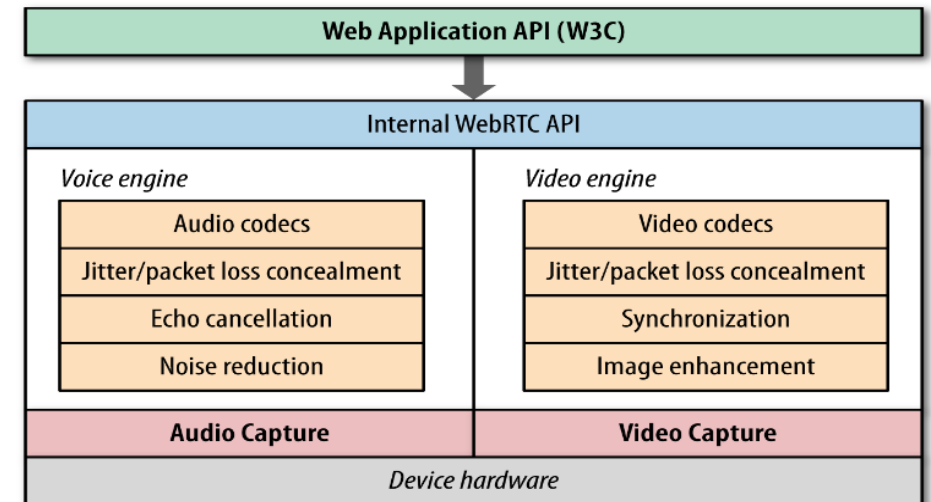
Browser  
■ Other ■ Electron ■ Firefox ■ Chrome

callstats.io/2018

# WebRTC API\*

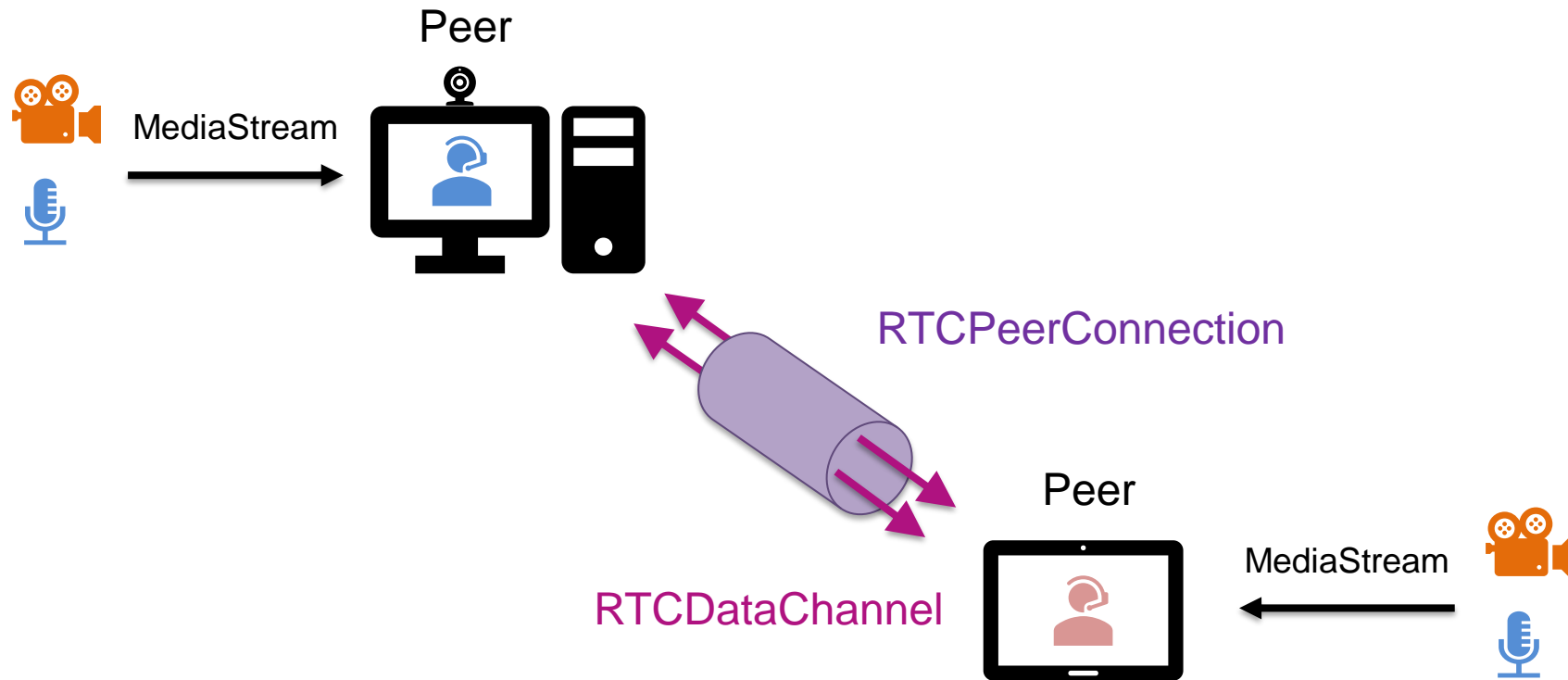
JavaScript APIs:

- 
**RTCPeerConnection**
  - Peer-to-Peer Verbindung
- 
**RTCDataChannel**
  - **bidirektionaler** Transfer von Daten zwischen Peers
  - Jeder Kanal ist mit **RTCPeerConnection verbunden**
  - Jeder Peer kann **mehr als einen Kanal verwenden**
- 
**MediaStream**
  - über die JavaScript Funktion "**getUserMedia**"
  - Ressourcen-Zugang/ Streams (Video, Audio, Data)

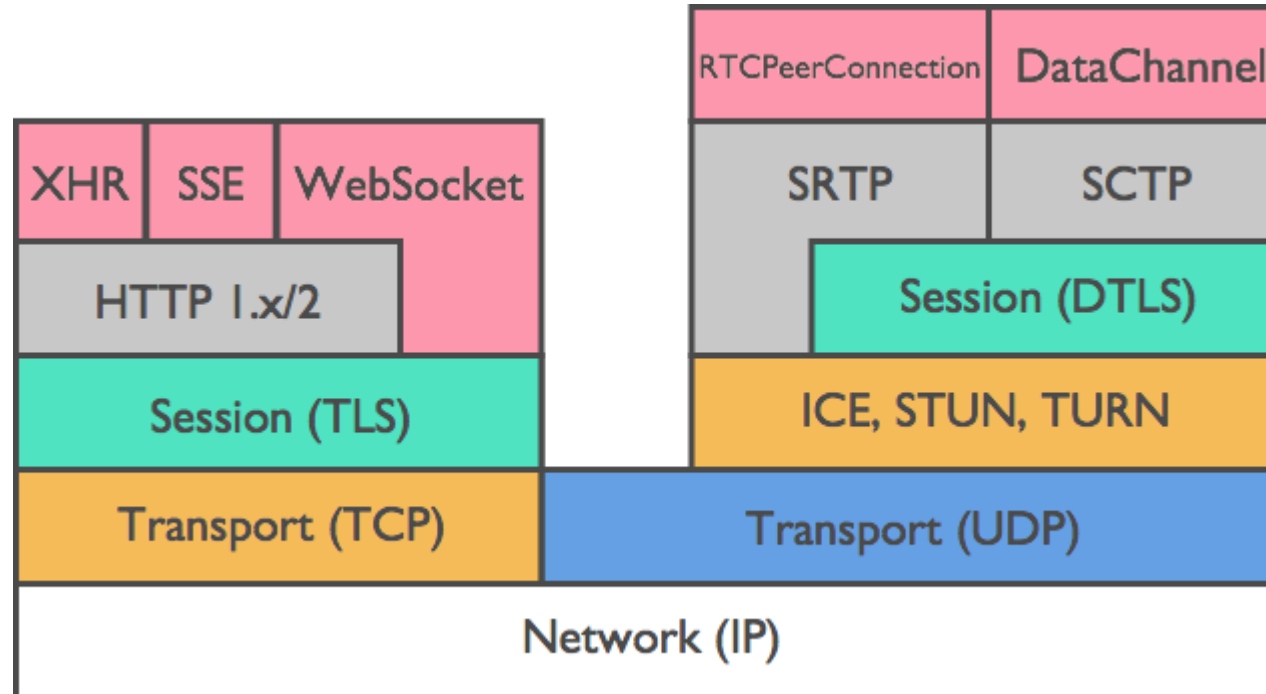


See G. Ilya, "High Performance Browser Networking", O'Reilly  
ISBN: 9781449344757. 2013

# WebRTC API Overview



# Stack



<http://webrtc-security.github.io/>

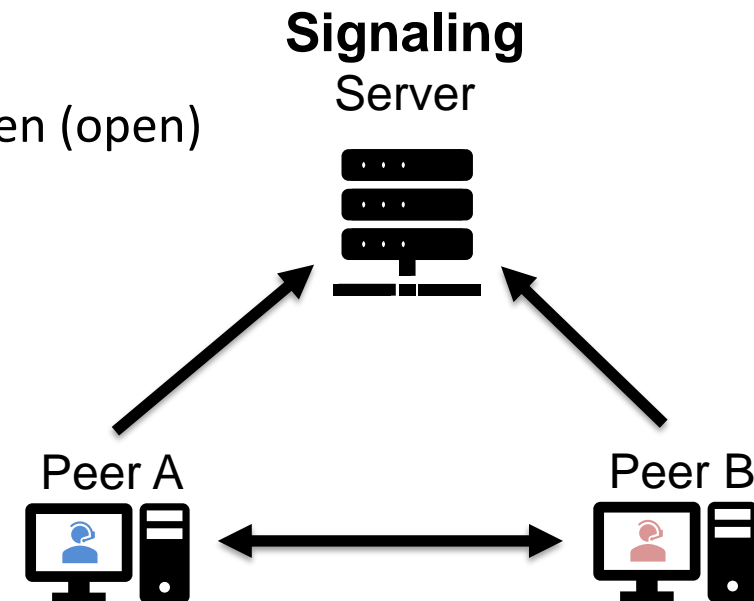
SRTP ... Secure Real-Time Transport Protocol (RFC 3711)

SCTP ... Stream Control Transmission Protocol (RFC 2960/4960)

DTLS ... Datagram Transport Layer Security (RFC 4347/6347)

# Verwendete Server

- 🌐 **Signaling Server**
  - Zwingend notwendig zur Vermittlung
  - Kann mit unterschiedlichen Technologien umgesetzt werden (open)
- 🌐 **NAT Traversal Servers**
  - mit geNATeten Netzwerken
  - In produktiven Umgebungen
- 🌐 **Media Server**
  - hängt von der Anwendung ab (**optional**)
- 🌐 **Gateway**
  - hängt von der Anwendung ab(**optional**)



# Signaling

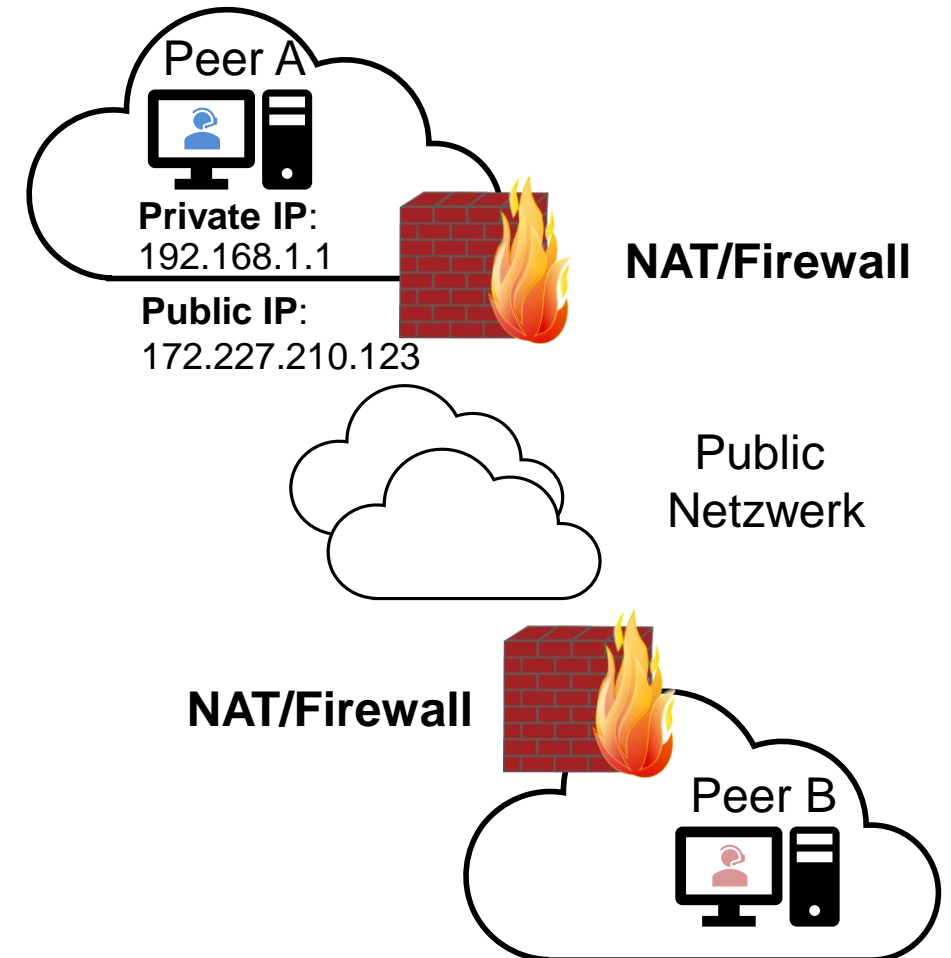
- 🌐 **Session Description Protocol (SDP)**
- 🌐 **Offer / Answer Prozess**
- 🌐 **Beschreibt, wie Daten gesendet werden**
- 🌐 **Security**
  - **Authentication** (Wer darf eine Sitzung starten?)
  - **Authorization** (Welche Benutzer dürfen was?)

# Signaling Server

- 🌐 **CPaaS** (Communication Plattform as a Service)
- 🌐 **Selbst hosten / Proprietär**
  - SimpleWebRTC
  - EasyRTC
  - GitHub
- 🌐 **Messaging Service**
  - Firebase
  - Google Cloud Messenger

# NAT Traversal Problem

- 🌐 Wenn zwei Peers miteinander sprechen wollen, brauchen sie ...
  - öffentliche IPs und
  - einen Port
- 🌐 Über **NAT und Firewalls** ist eine direkte Verbindung problematisch

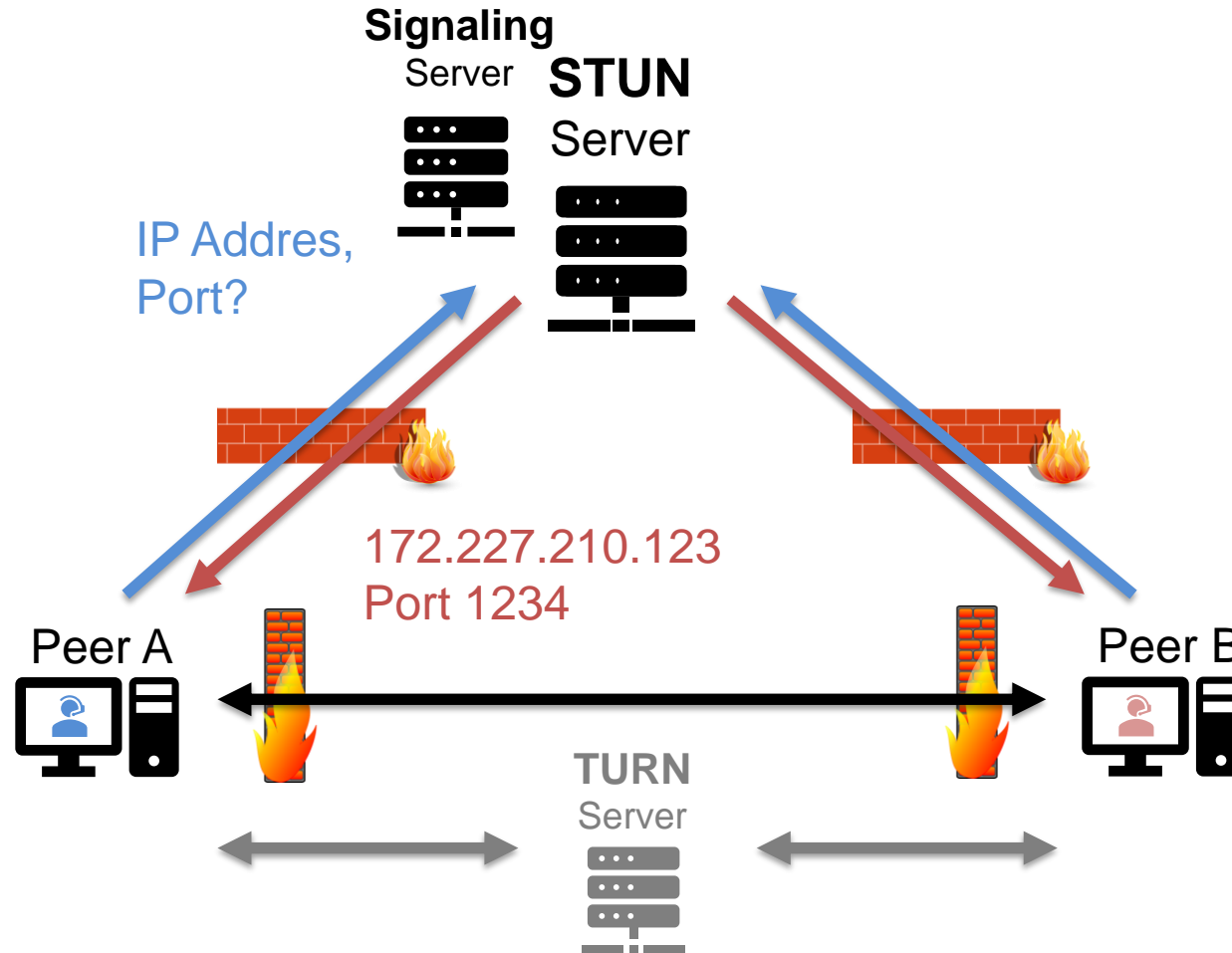




# ICE, STUN & TURN

- 🌐 Der **Interactive Connectivity Establishment (ICE)** Framework wird für den Aufbau von Verbindungen in **geNATeten Netzwerken** verwendet
- 🌐 Ein **Session Transversal Utilities for NAT (STUN)** Server erlaubt es den Peers, ihre öffentlichen **IPs und Ports und deren Typ von NAT** zu publizieren. Dadurch kann dann eine Peer-to-Peer Verbindung aufgebaut werden. (RFC 3489/5389)
- 🌐 In den meisten Fällen ist STUN ausreichend. Wenn keine Verbindung über **STUN** möglich ist, wird als Intermediär ein **Traversal Using Relay NAT (TURN) Server** zum Weiterleiten der Ströme eingesetzt. (RFC 5766)

# STUN & TURN



## STUN vs. TURN

	STUN Server	TURN Server
<b>Function</b>	Gibt externe IPs, Ports und NAT-Type bekannt	Traverse für Daten
<b>Wann gebraucht?</b>	Immer	Wenn keine Peer-to-Peer Verbindung über STUN
<b>Kosten (Ressourcen)</b>	Wenig	Teuer
<b>Qualität (Datenrate, ..)</b>	Keine	Möglich

# Handover der ICE Server (JSON)

```
{
  "iceServers": [
    {
      "urls": [
        "stun:turn-001-hstn.streamstack.io:3478"
      ]
    },
    {
      "urls": [
        "turn:turn-001-hstn.streamstack.io:3478?transport=udp",
        "turn:turn-001-hstn.streamstack.io:80?transport=tcp",
        "turns:turn-001-hstn.streamstack.io:443?transport=tcp"
      ],
      "credential": "XYZWkjEgSr3feHwgLUk28C6/MqA=",
      "username": "133783192:556fda624e8b4fdb333600fb"
    }
  ],
  "ttl": 3600
}
```

STUN server

TURN Servers  
UDP on standard STUN port  
TCP on standard web port  
TLS over TCP on HTTPS port

<https://www.youtube.com/watch?v=nRZePB4kzWo>

# Kandidaten für eine Audio Verbindung

## ICE servers

turn:numb.viagenie.ca [mathias.knoll@fh-joanneum.at]

Time	Component Type	Foundation	Protocol Address	Port	Priority
0.403	1 host	0	UDP 10.52.200.132	64241	126   32512   255
0.403	1 host	3	TCP 10.52.200.132	9	125   32704   255
0.404	2 host	0	UDP 10.52.200.132	64242	126   32512   254
0.404	2 host	3	TCP 10.52.200.132	9	125   32704   254
1.039	1 srflx	1	UDP 91.229.57.240	64241	100   32543   255
1.039	1 relay	2	UDP 158.69.221.198	58479	5   32543   255
1.061	2 srflx	1	UDP 91.229.57.240	64242	100   32543   254
1.061	2 relay	2	UDP 158.69.221.198	58480	5   32543   254
1.061					Done

**host** ... lokale Adresse

**srflx** ... (stun reflexive) Adresse von STUN Server

**relay** .. Adresse von TURN Server

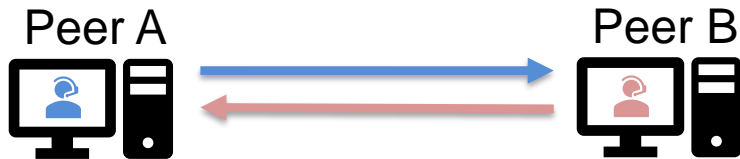
# ICE Kandidaten Protokolle

- 🌐 **UDP (turn)**
  - Nicht verlässlich
  - Für Streaming von Audio/Video
  - Ports könnten geblockt sein
- 🌐 **TCP (turn)**
  - Verlässlich
  - Nicht für Streams wegen Overhead
  - Üblicherweise nicht blockiert
- 🌐 **TLS über TCP (turns)**
  - Verlässlich
  - Not geeignet für Streams wegen (noch mehr) Overhead
  - Höchstwahrscheinlich blockiert
  - Doppelte Verschlüsselung  
(Kein Vorteil – nur da, um über den Firewall zu kommen)

# Multiple Users (Group Chat)

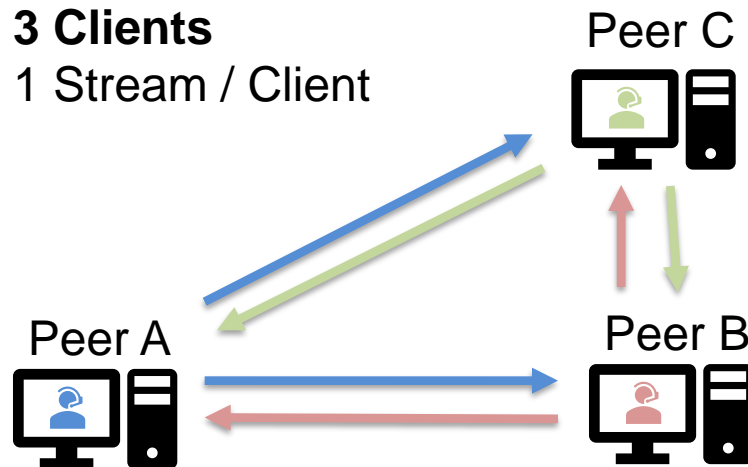
**2 Clients**

1 Stream / Client

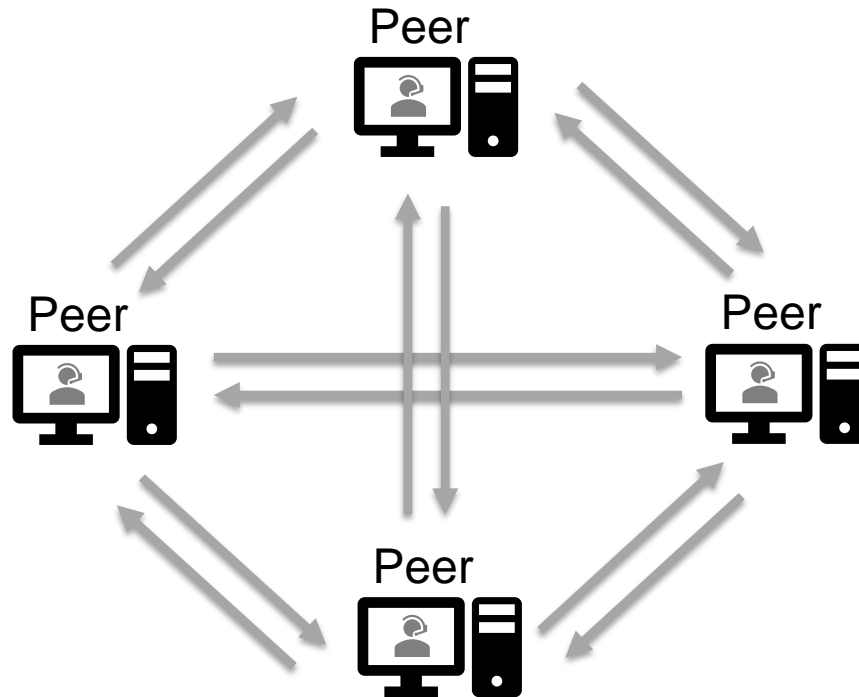


**3 Clients**

1 Stream / Client



**Scaling**



**Costs increase  
(Bandbreite)**

**4 Clients**

6 Streams per Client  
12 Streams in common

**n Clients**

**2(n-1) Streams / Client**  
**n<sup>2</sup>-n Streams in common**

**Solution:** Connect all of them together  
Workshop  
**WebSecurity**

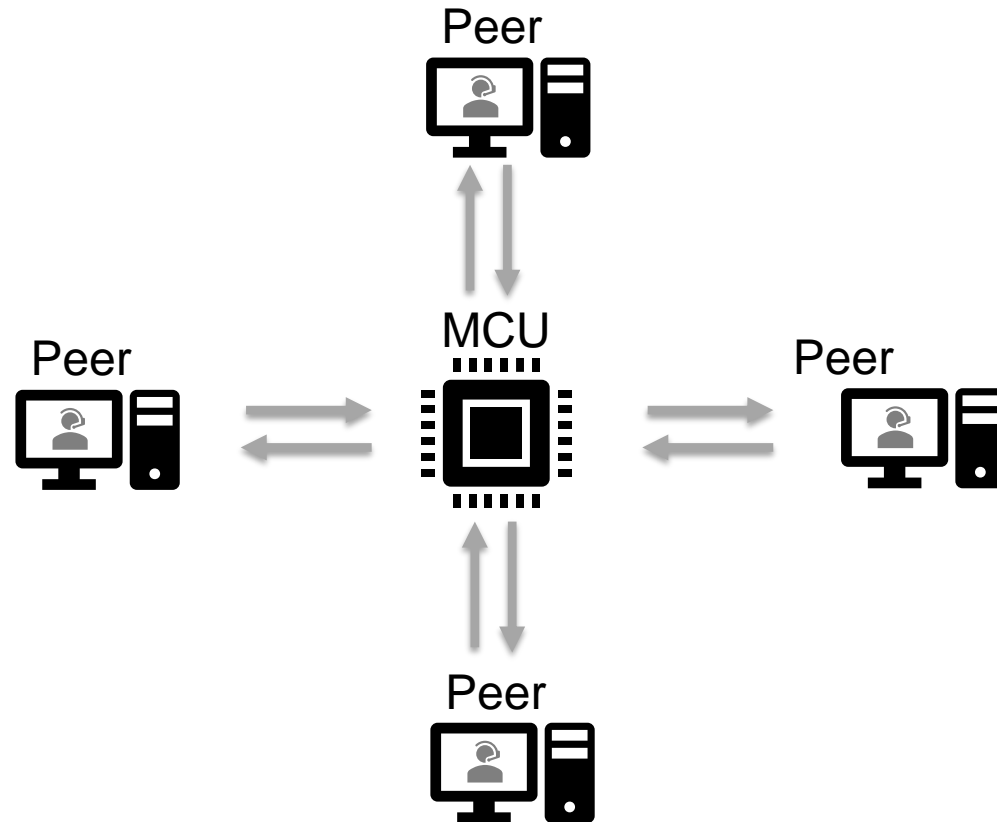
# Multipoint Control Unit

## 4 Clients

2 Streams per Client  
8 Streams in common

## n Clients

2 Streams / Client  
2n Streams in common



## Advantage:

Easy for Client

## Disadvantages:

MCU CPU & Data Rate costs

It is possible that Clients receive their own stream

Stream in return

Latency increases

Not flexible



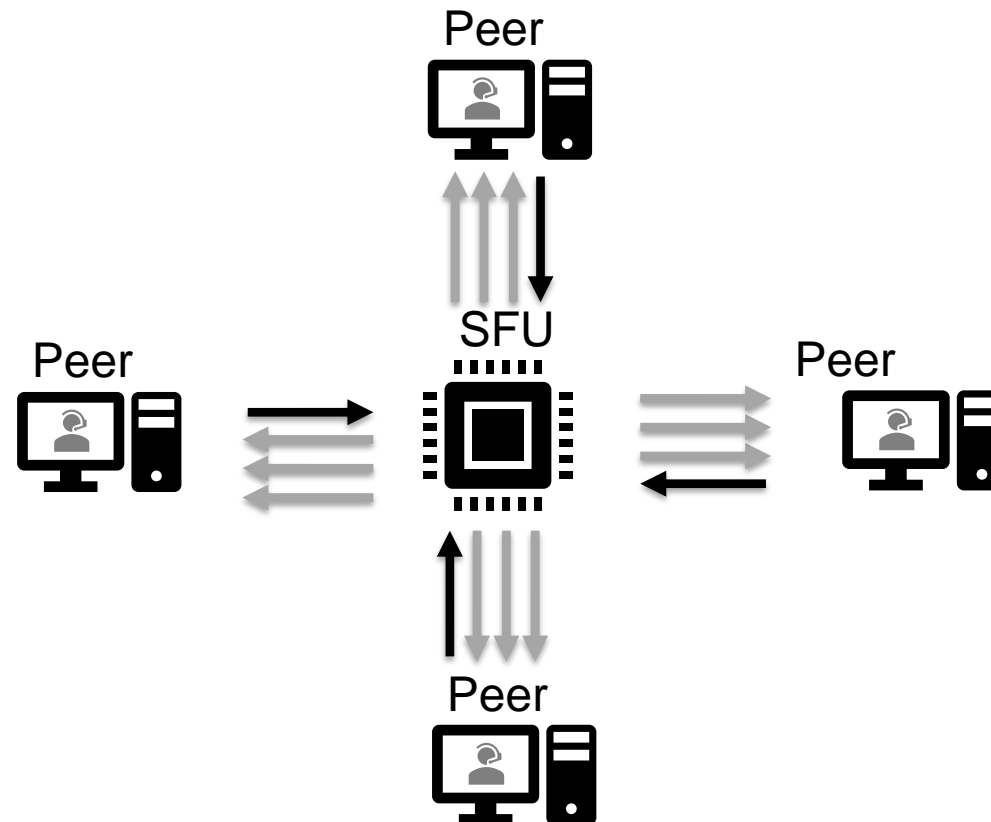
# Selective Forwarding Unit

## 4 Clients

4 Streams per Client  
16 Streams in common

## n Clients

n Streams / Client  
 $n^2$  Streams in common



## Advantages:

Skalierbare Server  
Infrastruktur  
Wenig Latenz

## Disadvantages:

MCU CPU & Data Rate  
costs  
Clients have to process a  
lot (mobile devices =  
batteries may drain faster!)

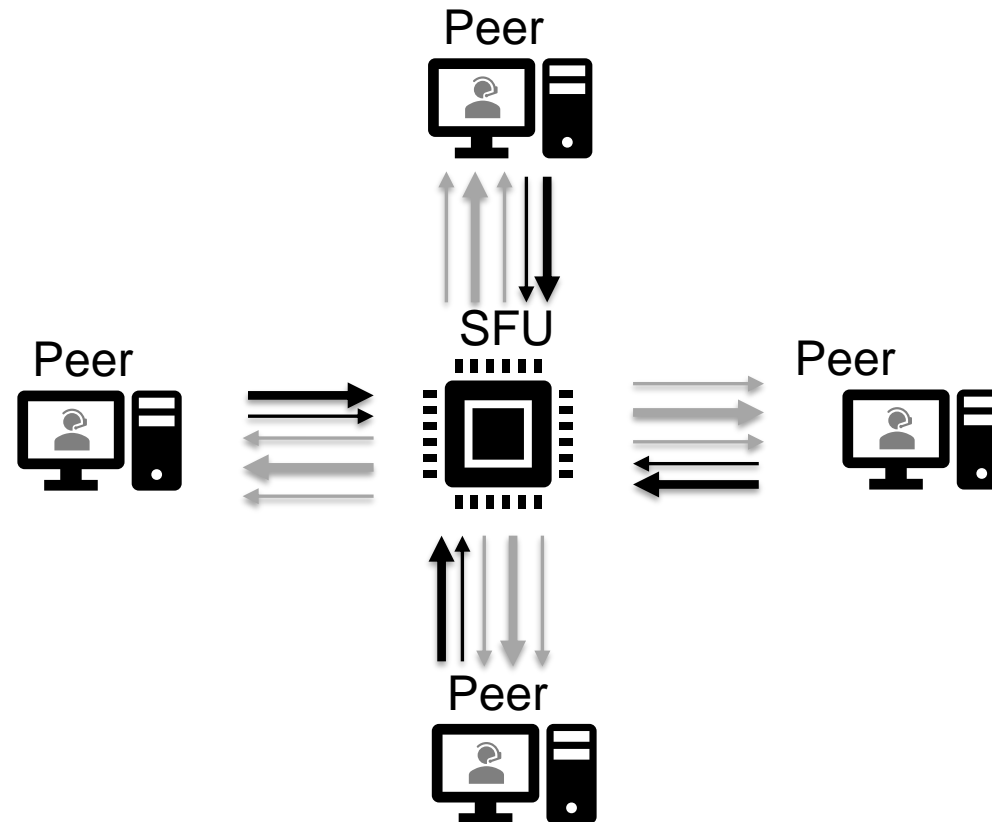
# SFU mit Simulcast

## 4 Clients

5 Streams per Client  
20 Streams in common

## n Clients

$n+1$  Streams / Client  
 $n^2+n$  Streams in common








## Advantages:

- Scalable Server Infrastructure
- Little latency
- Efficient processing (batteries!)

## Disadvantages:

- Complex
- Not standardized

## URLs

-  <https://webrtc.org/>
-  <https://w3c.github.io/webrtc-pc/>
-  <https://webrtcchacks.com/sdp-anatomy/>
-  <https://codelabs.developers.google.com/codelabs/webrtc-web/#0>
-  <http://webrtc-security.github.io/>