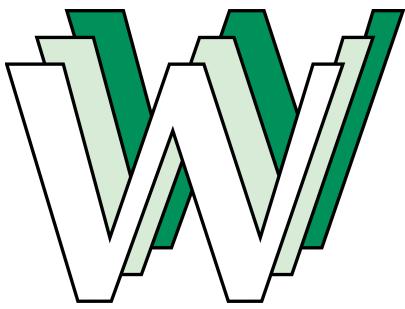


# **Grundlagen "Web-Engineering"**

# World Wide Web



"Das **World Wide Web** (englisch für „weltweites Netz“, kurz Web oder WWW) ist ein über das **Internet** abrufbares System von elektronischen Hypertext-Dokumenten, sogenannten **Webseiten**, welche mit **HTML** beschrieben werden.

Sie sind durch **Hyperlinks** untereinander verknüpft und werden im Internet über die Protokolle **HTTP** oder **HTTPS** übertragen.

Der Benutzer kann dann den **Hyperlinks** auf der angezeigten Webseite folgen, die auf andere Webseiten verweisen. So ergibt sich ein weltweites Netz aus Webseiten. Das Verfolgen der Hyperlinks wird auch als „**Surfen im Internet**“ bezeichnet.

Die Webseiten enthalten meist **Texte**, oft mit **Bildern und grafischen Elementen** illustriert.  
Häufig sind auch **Videos, Tondokumente oder Musikstücke** eingebettet."

Zum Aufrufen von Inhalten aus dem World Wide Web wird ein **Webbrowser** benötigt, der z. B. auf einem **PC** oder einem **Smartphone** läuft. Mit ihm kann der Benutzer die auf einem beliebigen, von ihm ausgewählten **Webserver** bereitgestellten Daten herunterladen und auf einem geeigneten Ausgabegerät wie einem **Bildschirm** oder einer **Braillezeile** anzeigen lassen.

(vgl. [https://de.wikipedia.org/wiki/World\\_Wide\\_Web](https://de.wikipedia.org/wiki/World_Wide_Web))

- "Hyperlinks" / URL
- Client - Server
- HTTP
- Webbrowser
- PC / Smartphone
- HTML
- Texte, Bilder, Video, Ton

# **Engineering**

Engineering is the practice of using **natural science, mathematics**, and the **engineering design process** to solve **technical problems**, increase **efficiency and productivity**, and **improve systems**.

The **engineering design process** is a common series of steps that engineers use in creating **functional products and processes**. The process is **highly iterative**[...]

It is a **decision making process** in which the **basic sciences** [...] are applied to convert resources optimally to meet a stated objective.

Among the fundamental elements of the design process are the **establishment of objectives and criteria, synthesis, analysis, construction, testing and evaluation**.

(vgl. <https://en.wikipedia.org/wiki/Engineering>,  
[https://en.wikipedia.org/wiki/Engineering\\_design\\_process](https://en.wikipedia.org/wiki/Engineering_design_process))

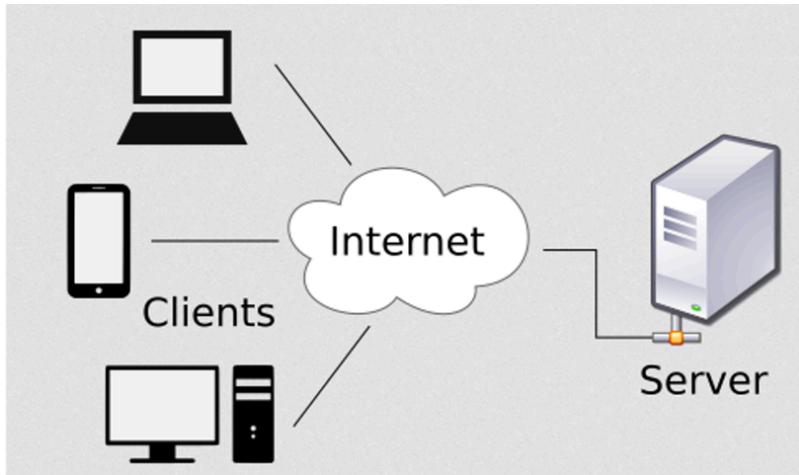
- **Anwendung von Wissenschaften**
- Lösen von technischen Problemen, erhöhen der Effizienz, Optimierung von Systemen
- **Iterativer Prozess**
- Entscheidungsverfahren
- Bestimmen der Anforderungen und **Analyse, Herstellung, Überprüfung und Evaluation**

## "Hyperlinks" / URL

<https://www.philipackermann.de:80/books/web.html?language=de#chapter7>

## Client - Server

- Client: Browser -> Eher einfache Maschine (Software und Hardware), meistens in direktem Zugriff des Users
- Server: Webserver -> eher leistungsfähig (Software und Hardware), i.d.R. im Rechenzentrum



# HTTP

```
$ telnet www.perdu.com 80
Trying 208.97.177.124...
Connected to www.perdu.com.
Escape character is '^]'.

GET / http/1.1
Host: www.perdu.com

HTTP/1.1 200 OK
Date: Sat, 17 Aug 2013 12:14:56 GMT
Server: Apache
Accept-Ranges: bytes
X-Mod-Pagespeed: 1.1.23.1-2169
Vary: Accept-Encoding
Cache-Control: max-age=0, no-cache
Content-Length: 204
Content-Type: text/html

<html><head><title>Vous Etes Perdu ?</title></head><body><h1>Perdu sur l'Internet ?</h1><h2>Pas de panique, on va vous aider</h2><strong><pre> * <----- vous &ecirc;tes ici</pre></strong></body></html>
Connection closed by foreign host.
$
```

*Verbindungsaubau zum Server*

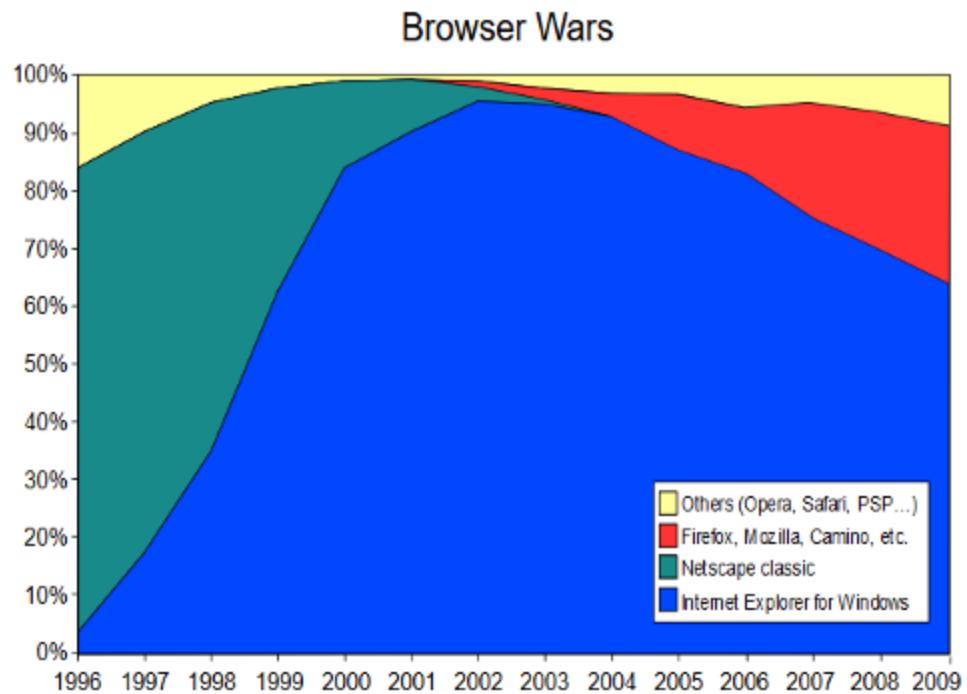
*HTTPAnfrage*

*Serverantwort: Header*

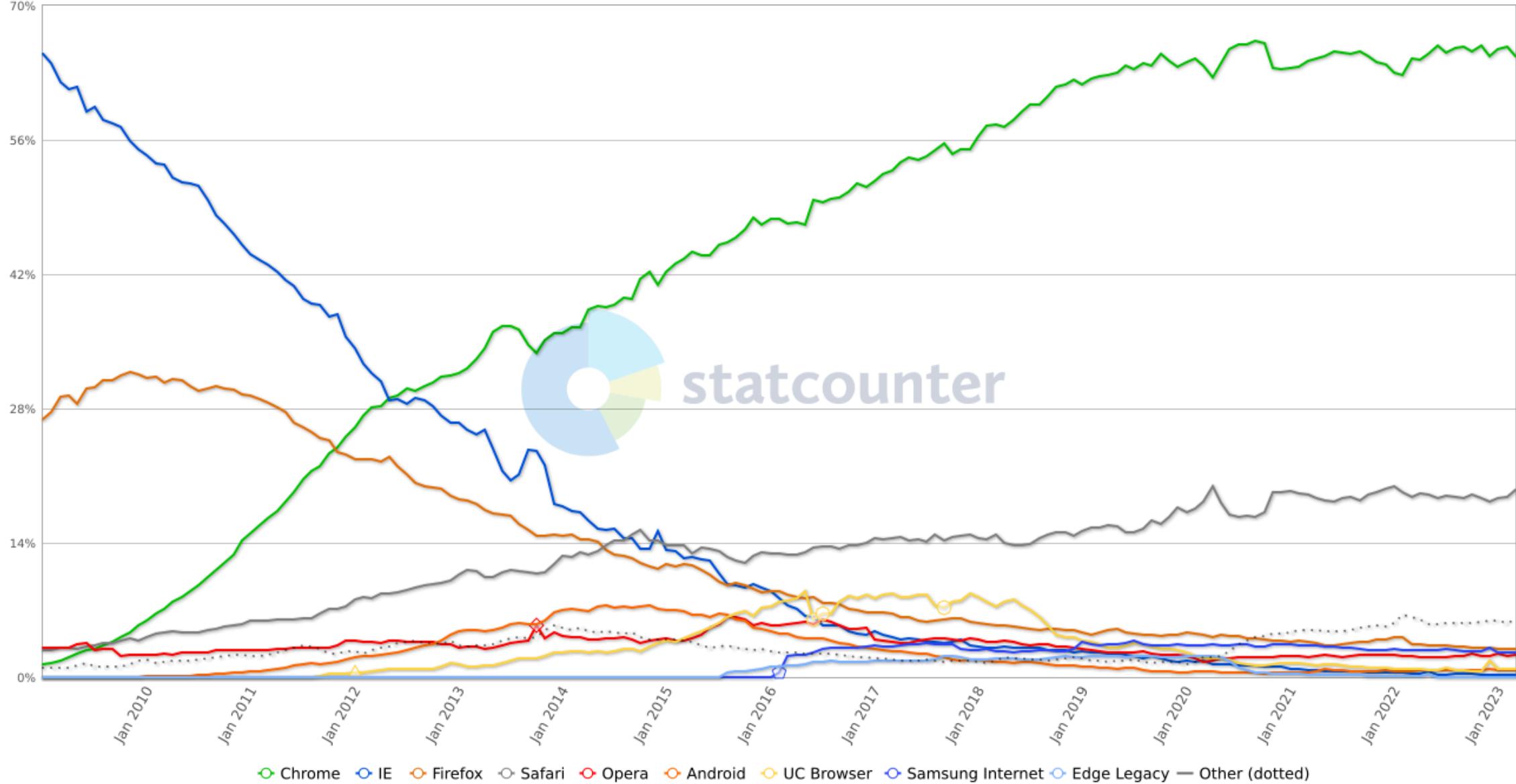
*Serverantwort: Nachrichtenrumpf*

*Verbindungsende*

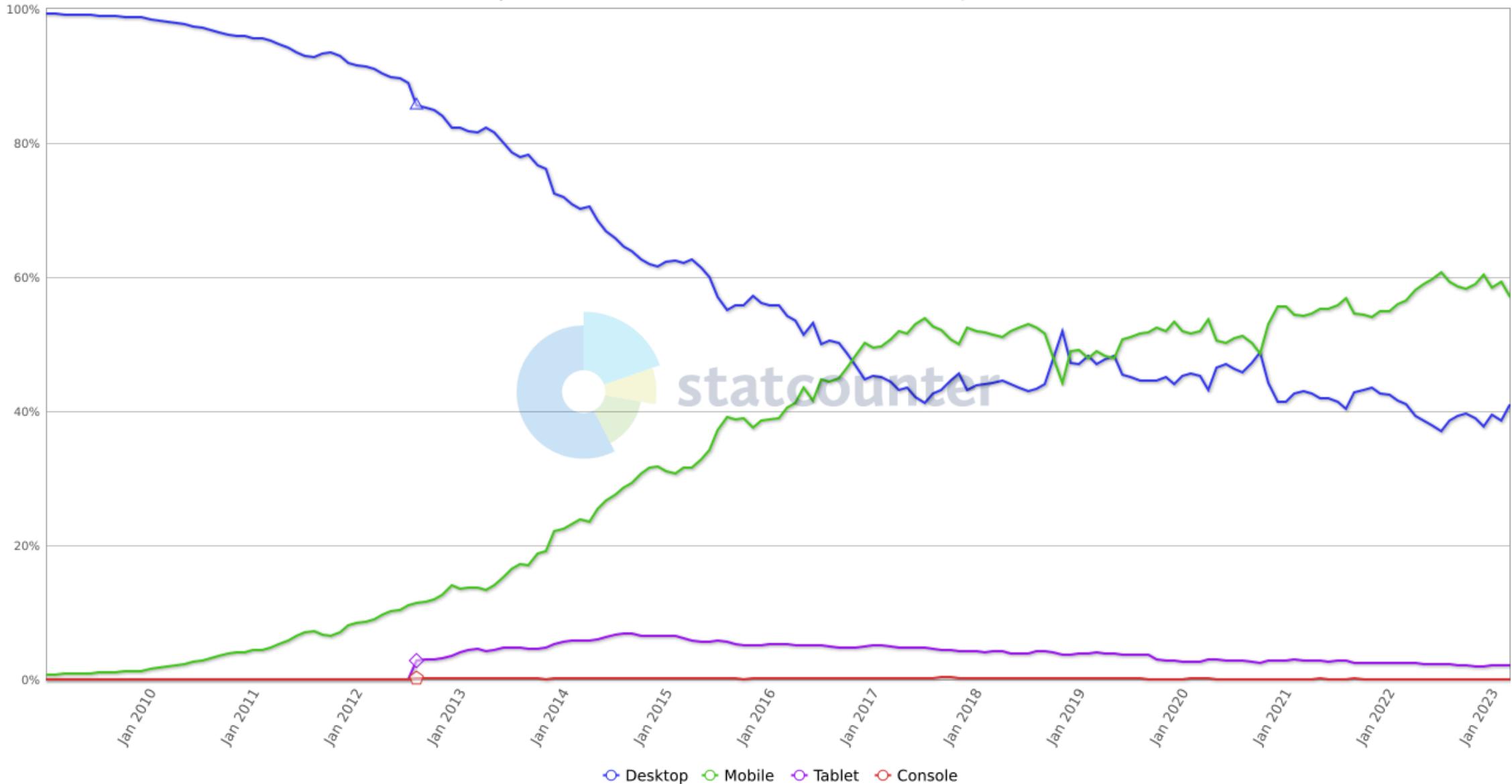
# Webbrowser



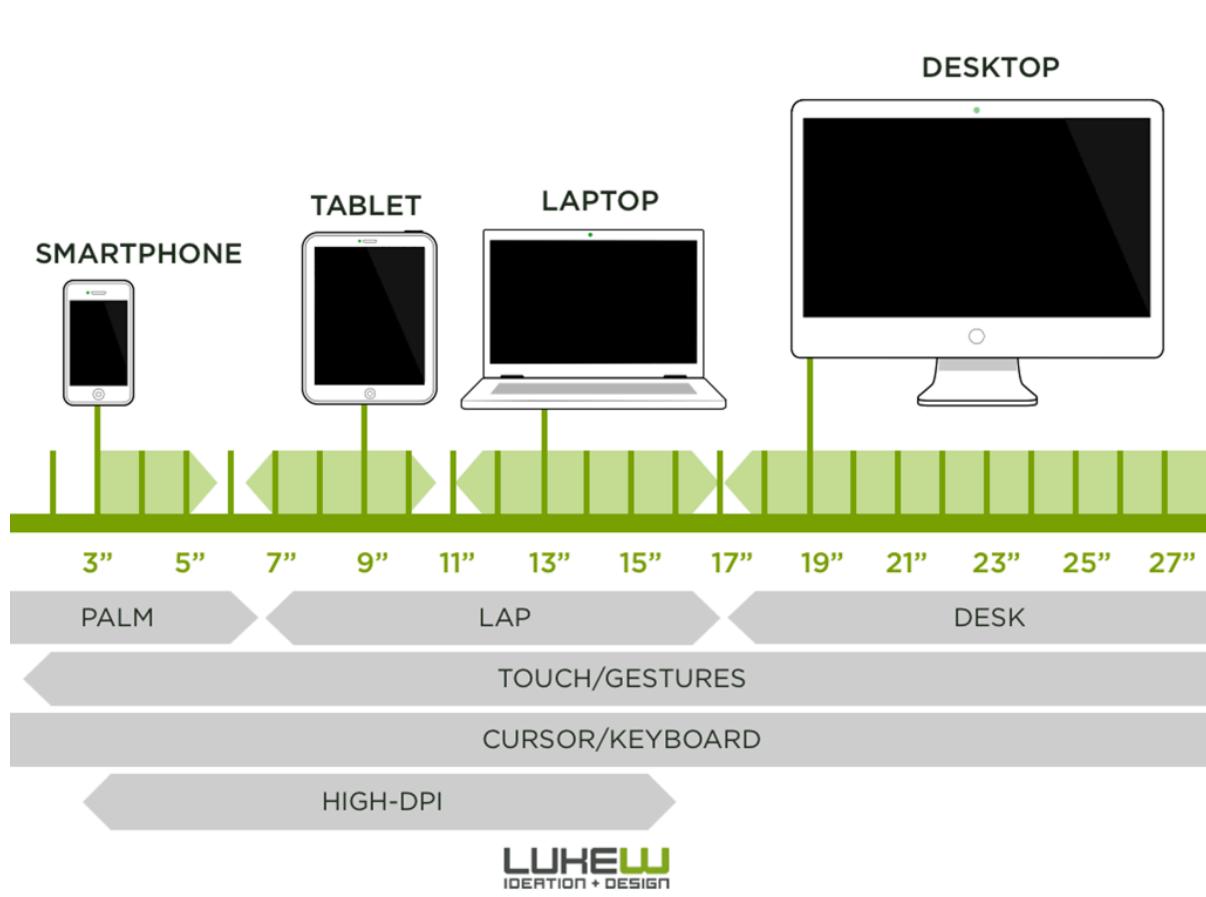
**StatCounter Global Stats**  
Browser Market Share Worldwide from Jan 2009 - Mar 2023



**StatCounter Global Stats**  
Desktop vs Mobile vs Tablet vs Console Market Share Worldwide from Jan 2009 - Mar 2023



# PC / Smartphone

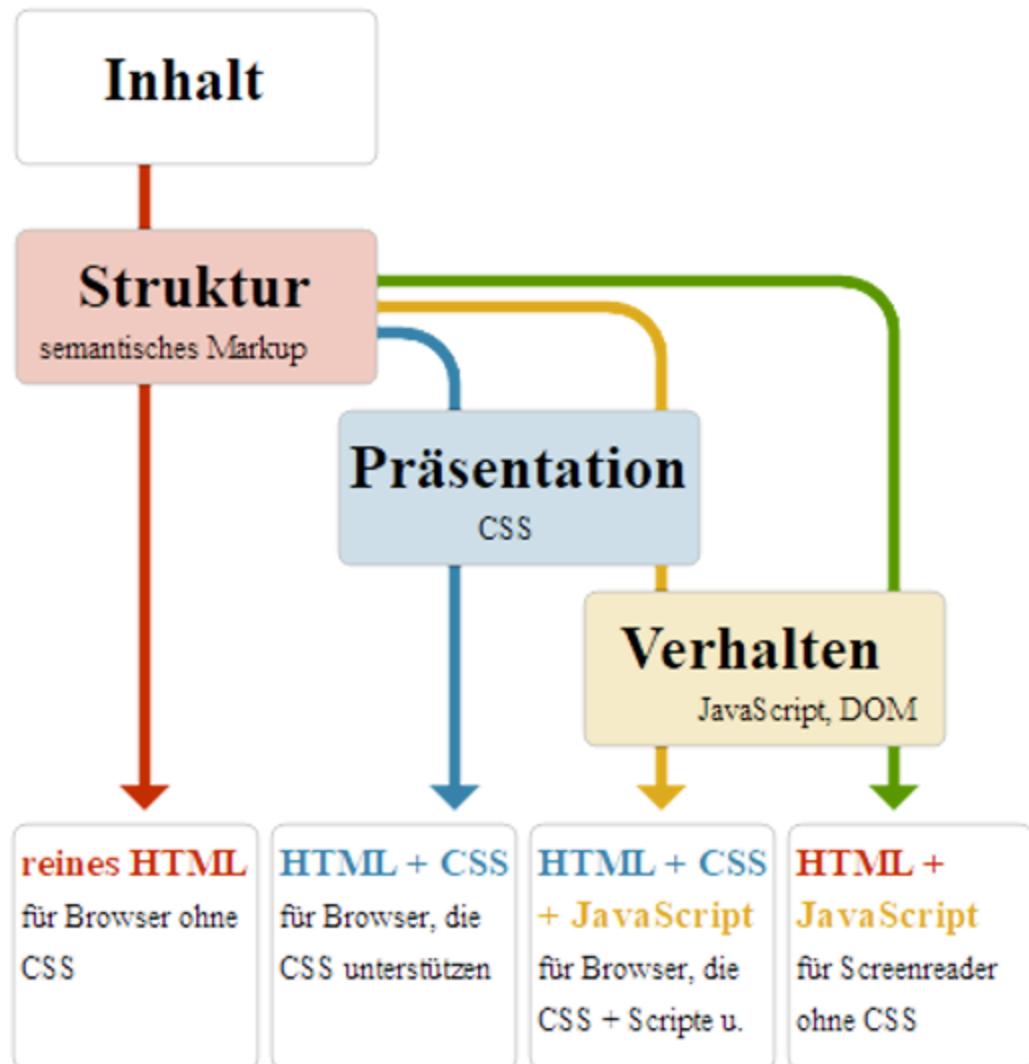


# Webseiten strukturieren mit HTML

```
<!DOCTYPE html>
<html lang="de">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Titel</title>
</head>
<body>

</body>
</html>
```

# Aufbau von Webapplikationen



# **Software Stacks**

# LAMP

## *LAMP Stack*



Linux



Apache



MySQL



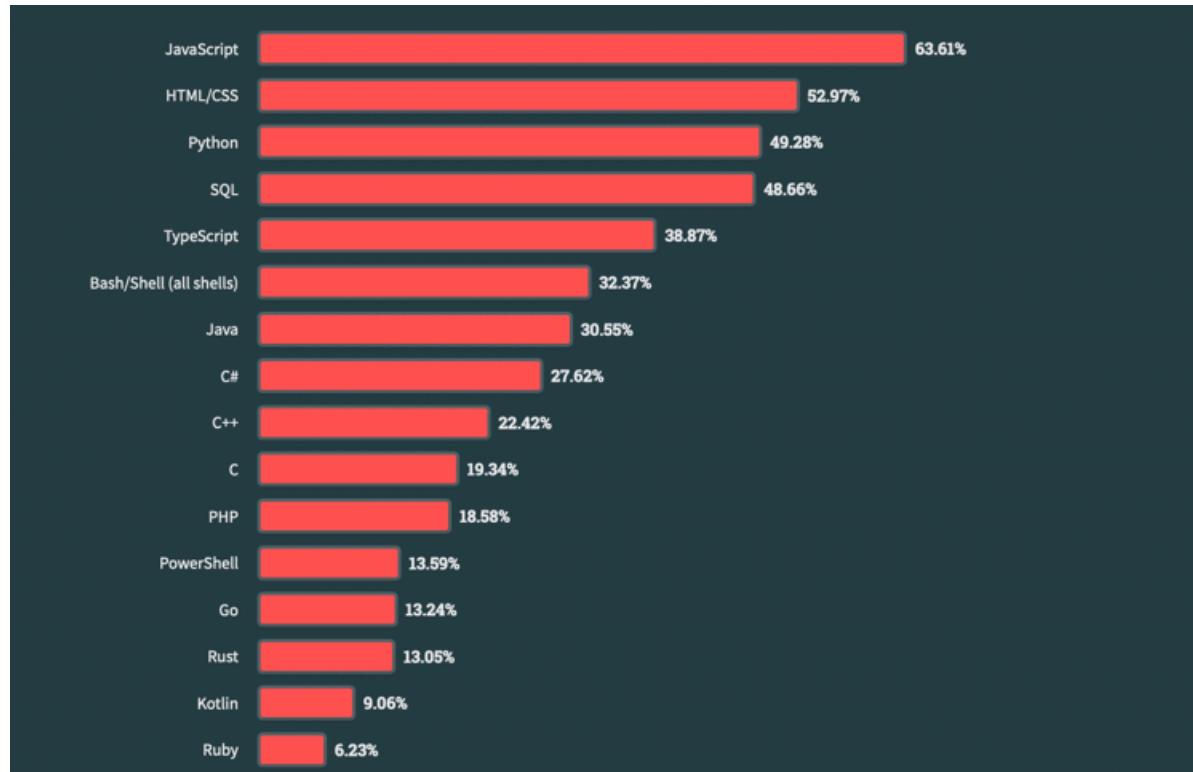
PHP

# MERN

MongoDB, Express.js, React, Node.js

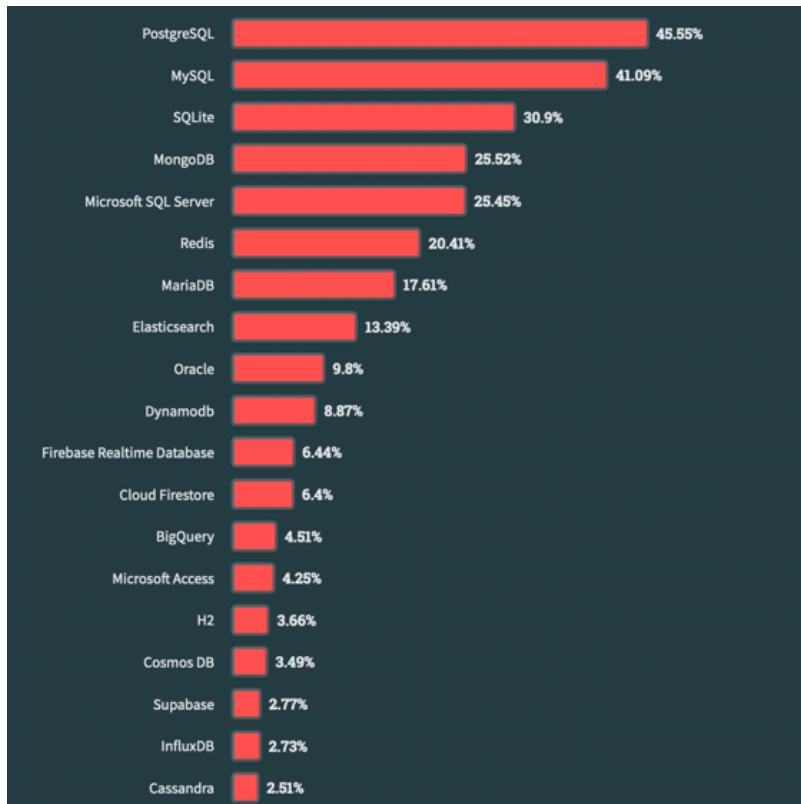


# Programmiersprachen



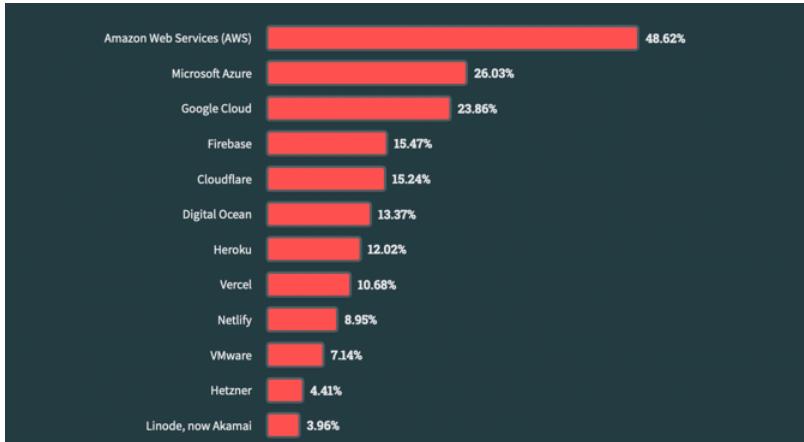
<https://survey.stackoverflow.co/2023/#section-most-popular-technologies-programming-scripting-and-markup-languages>, 23.04.24

# Datenbanken



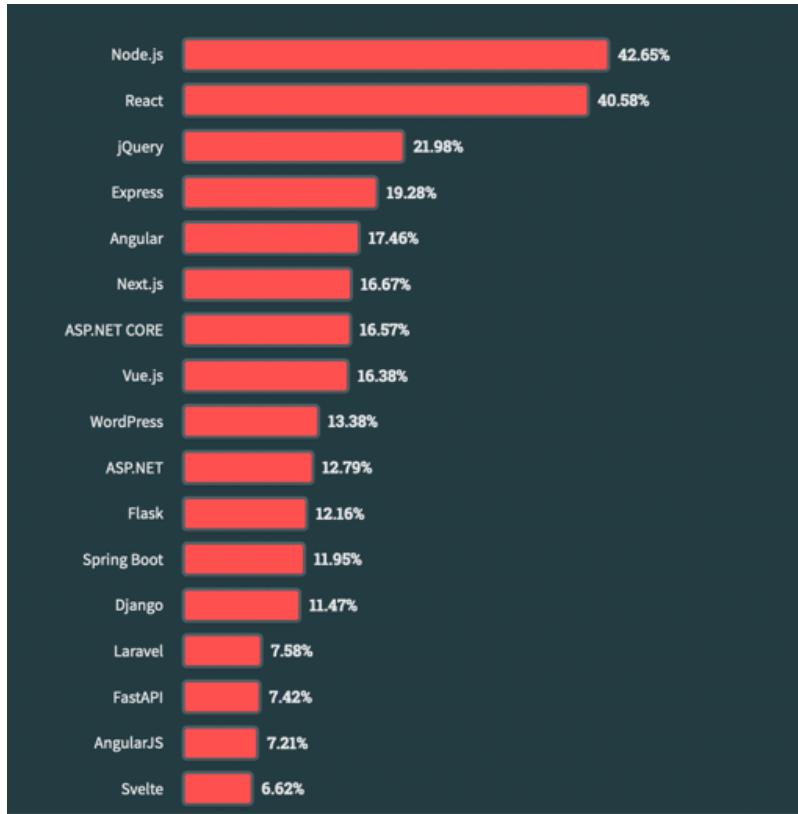
<https://survey.stackoverflow.co/2023/#section-most-popular-technologies-databases>,  
23.04.24

# Cloud Plattformen



<https://survey.stackoverflow.co/2023/#section-most-popular-technologies-cloud-platforms>, 23.04.24

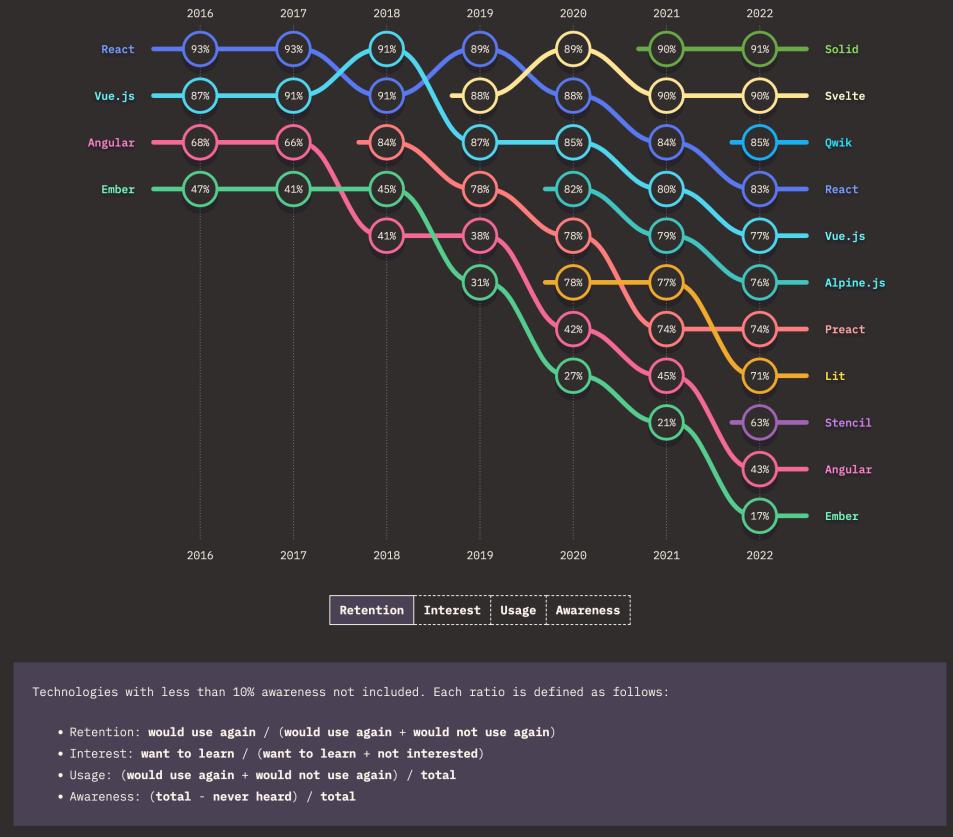
# Web Frameworks



<https://survey.stackoverflow.co/2023/#section-most-popular-technologies-web-frameworks-and-technologies>, 23.04.24

## RANKINGS OVER TIME

Retention, interest, usage, and awareness ratio rankings.

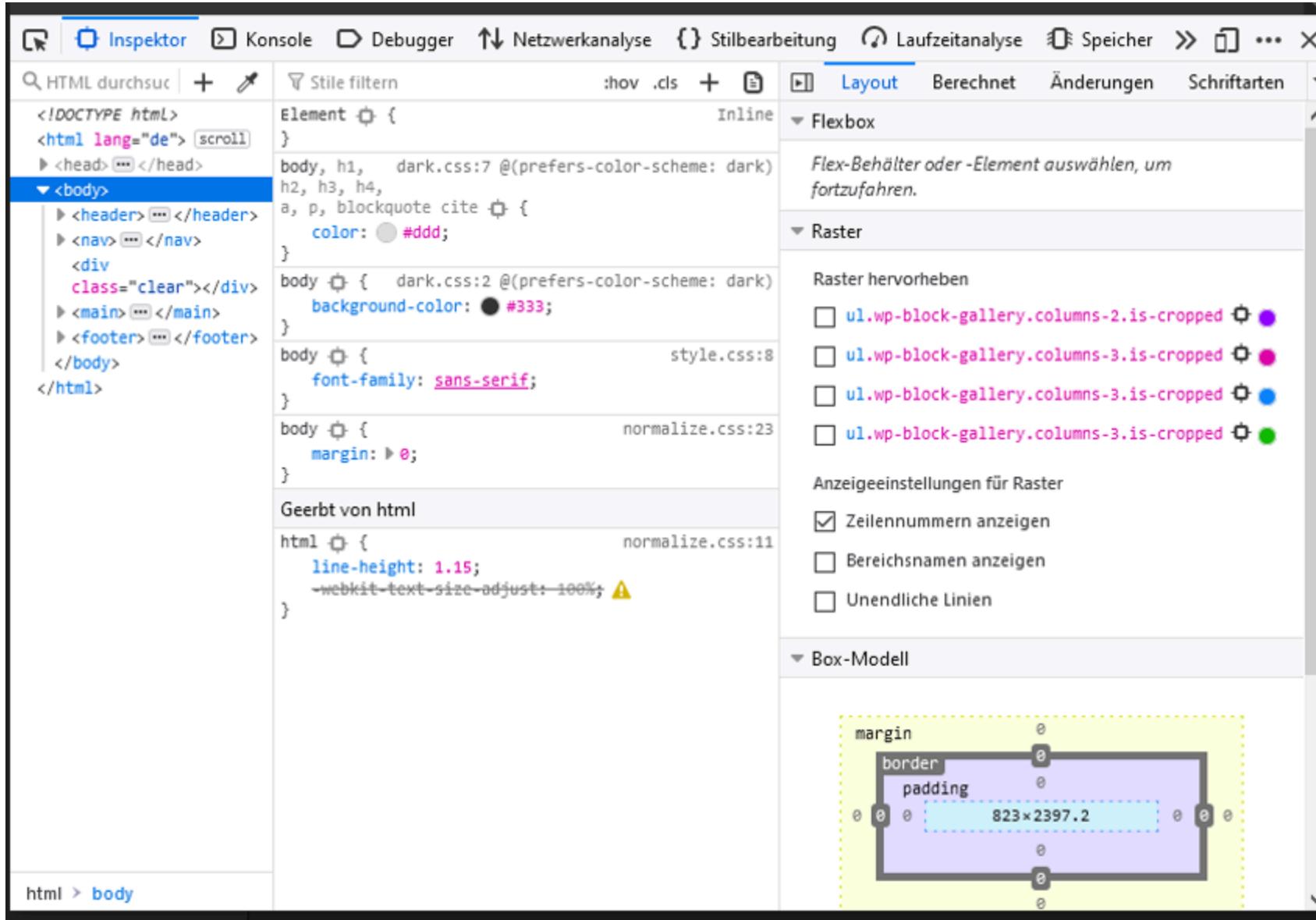


State of JavaScript 2022, 21.06.2023

**Texte, Bilder, Video, Ton**

# Werkzeuge

## Tools.md

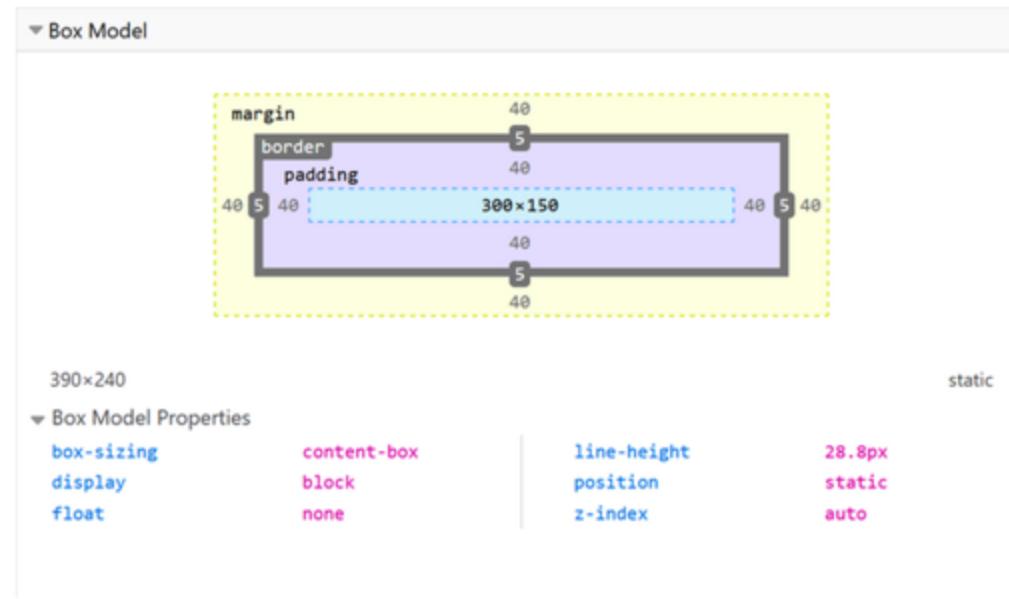


# **Webseiten gestalten mit CSS**

# Layoutkonzepte

- <http://info.cern.ch/hypertext/WWW/TheProject.html>
- Framesets
- Tabellen
- Cascading Style Sheets (CSS)
- Fixed vs. Liquid Layout
- Responsive Webdesign
- Device Agnostic
- Mobile First

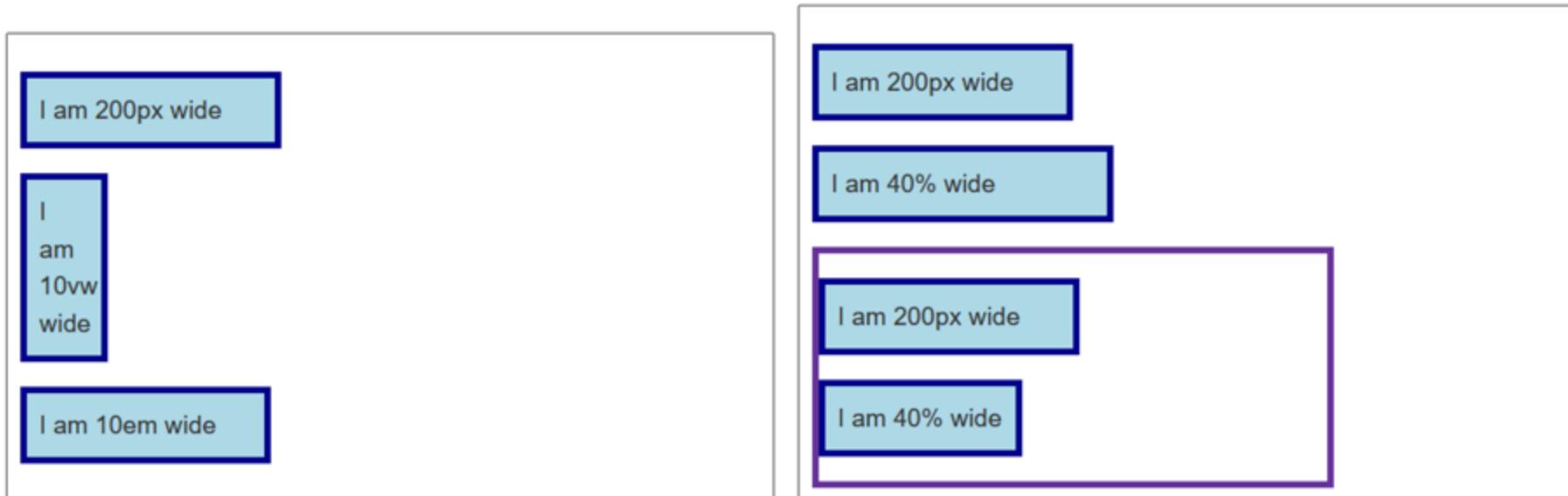
# Box Model



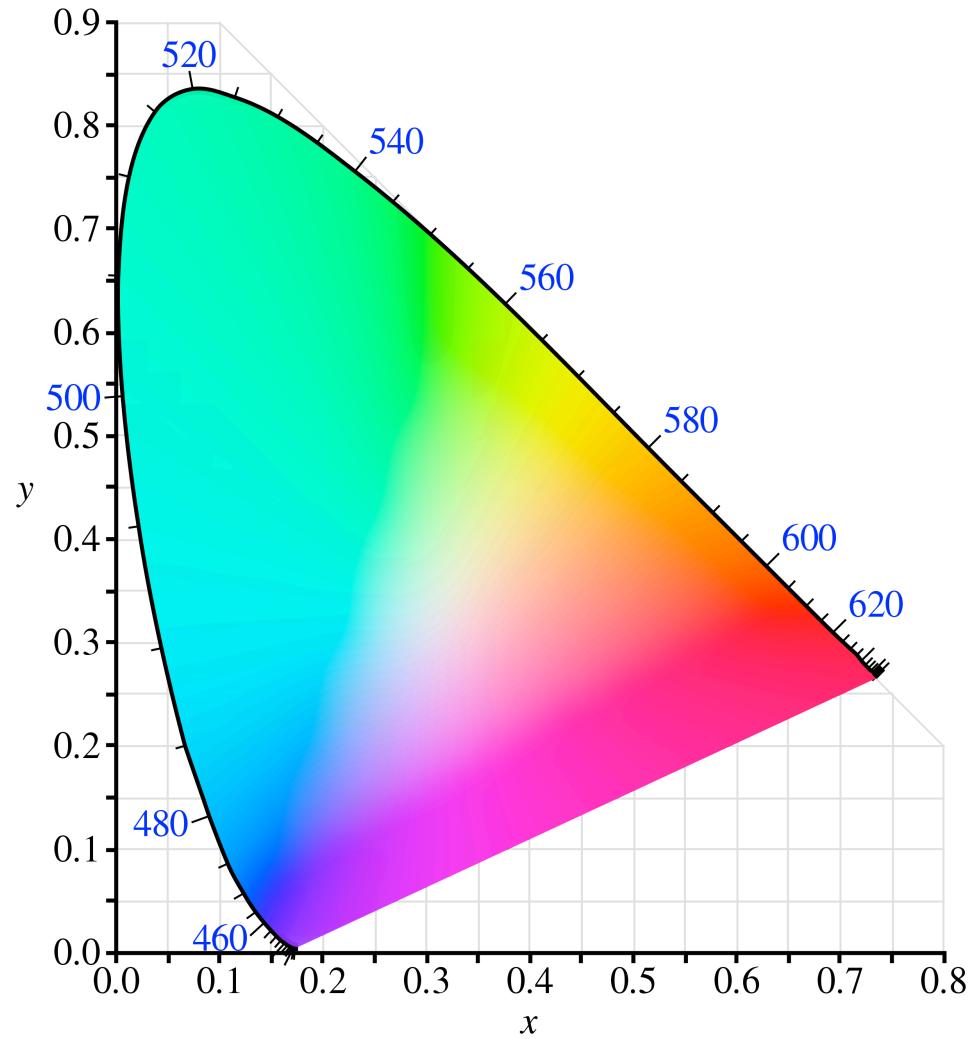
- `box-sizing: content-box` : `width` bezieht sich nur auf den content (blau,  $300 \times 150$  )
- `box-sizing: border-box` : `width` bezieht sich auf content + padding + border (blau, violett, grau,  $300+2*40+2*5$  für die Breite)

# Einheiten

- Absolute Größen: px ( cm , mm , ...) -> sparsam verwenden
- Relative Größen
  - em : Schriftgrösse des Elternelements
  - rem : Schriftgrösse des Wurzelelements
  - vw , vh : viewport breite, viewport höhe

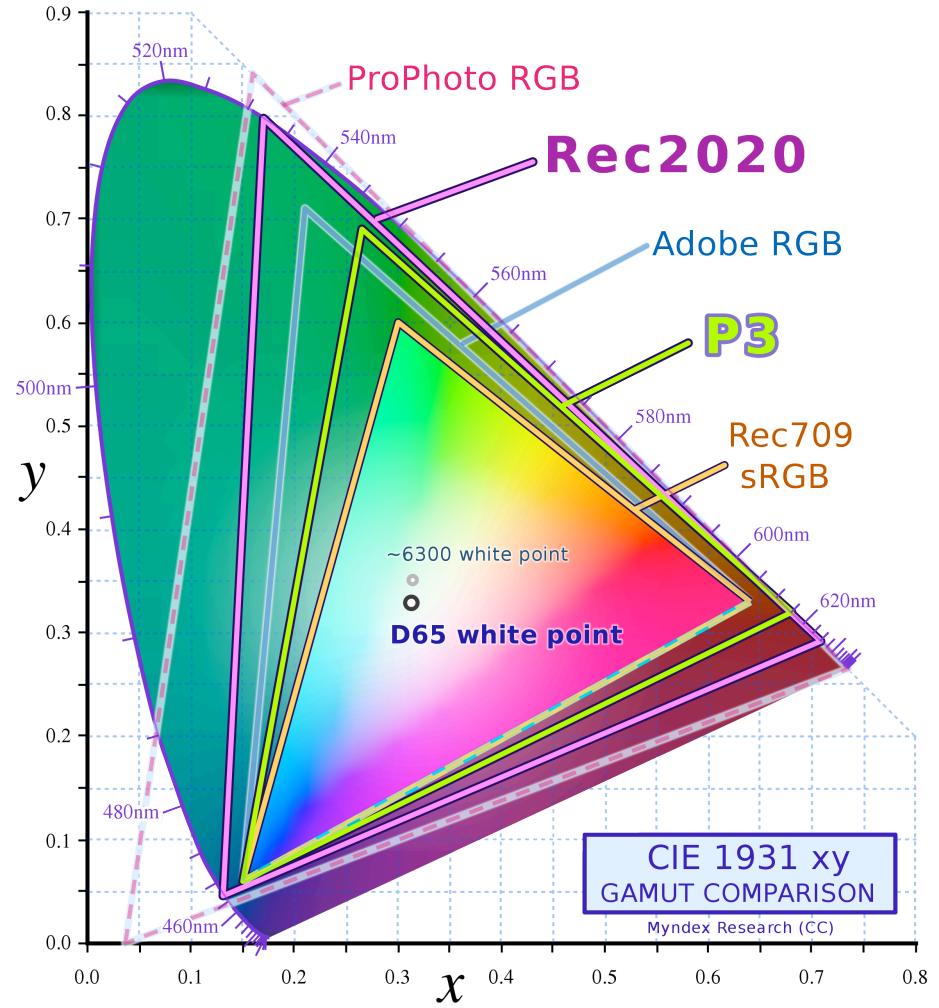


# Farben



CIE 1931 Farbraum

# Vergleich Farbräume



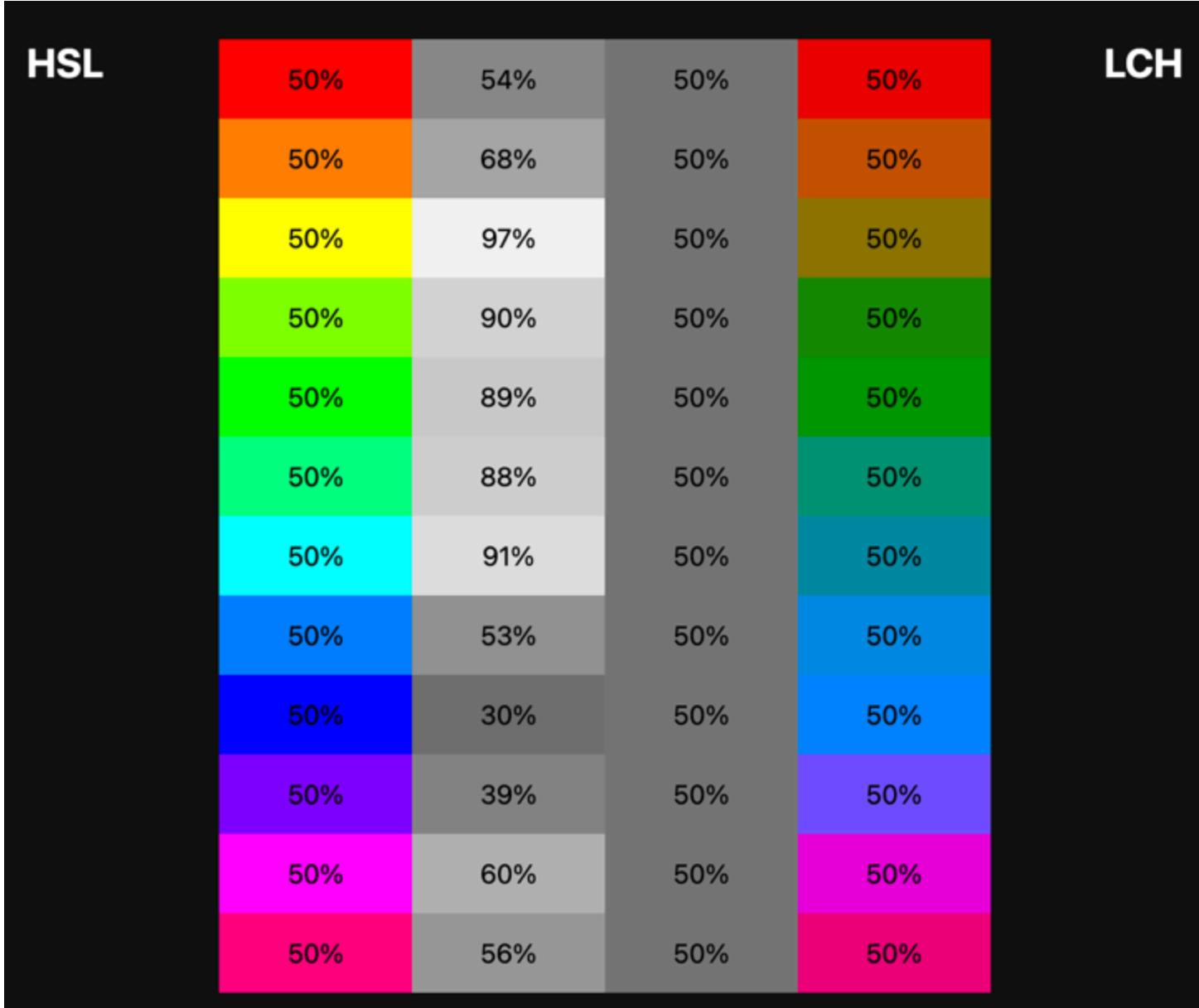
# **Farben in CSS**

## sRGB Farbraum

- Farbnamen: `color: darkblue;`
- Hex-Werte: `color: #ffa500;`
- RGBA-Werte (mit Deckkraft): `color: rgba(169, 169, 169, 0.5)`
- HSL-Werte (Hue, Saturation, Lightness): `color: hsl(60, 100, 50)`

## Alle sichtbaren Farben

- LCH (Lightness Chroma Hue / Opacity): `color: lch(29.2345% 44.2 27 / 0.5)`
- Oklch: `color: oklch(40.1% 0.123 21.57)`
- CIELAB (Lightness, red-green, blue-yellow): `color: lab(29.2345% 39.3825 20.0664);`
- Oklab: `color: oklab(40.1% 0.1143 0.045);`



<https://codepen.io/web-dot-dev/pen/poZgXxy>

# **Webseiten interaktiv machen mit JavaScript**

vgl.: Douglas Crockford (2018): How JavaScript Works, virgule solidus

## How Class Free Works

- Klassen sind syntaktischer Zucker, d.h. sie bieten keine Funktionalität, die nicht mit anderen Mitteln erreicht werden kann.
- Sie verhalten sich anders als Klassen in C++, Java oder C#. Das kann verwirrend sein.

## "Composition over Inheritance"

- Vererbung ist weniger zentral als manche Sprachen oder Kurse vermitteln.
- Vererbung bringt auch einige Probleme mit sich, da die Klassen sehr eng gekoppelt sind und nicht explizit klar ist, welche Methoden aufgerufen werden.
- Komposition ist sehr leistungsfähig.

## Closures

Verschachtelte Funktionen können auf Variablen aus den äusseren Funktionen zugreifen.  
Auch nach deren Ausführung.

```
function init() {  
  var name = "Mozilla"; // name is a local variable created by init  
  function displayName() {  
    // displayName() is the inner function, that forms the closure  
    console.log(name); // use variable declared in the parent function  
  }  
  displayName();  
}  
init();
```

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Closures>

## Folgende Struktur wird empfohlen:

```
function counter_constructor() {
    // private property
    let counter = 0;

    // composition
    const reuse = other_constructor();

    function up() {
        counter -= 1;
        return counter;
    }

    function down() {
        counter += 1;
        return counter;
    }

    // freeze to make the object immutable
    return Object.freeze({
        // make functions up and down public
        up,
        down,
        // expose goodness property from another object
        goodness: reuse.goodness
    })
}
```

# Asynchronität

- JavaScript wurde primär für User-Interaktionen entwickelt.
- Asynchronität ist deshalb ein zentrales Sprachfeature.
- Es gibt verschiedene Möglichkeiten für asynchronen Code:
  - Callbacks
  - Promise
  - `async / await`

`async / await` ist verwirrend, weil damit Code produziert wird, der synchron aussieht, aber asynchron funktioniert.

## Callback-Funktionen

- Callback-Funktionen werden als Parameter einer Funktion übergeben und von dieser aufgerufen.
- Die sogenannte "Callback-Hell", gemeint ist die Verschachtelung von Callbacks in Callbacks, sollte vermieden werden.

```
function foo(callback) {  
    // some functionality  
  
    callback(value);  
}  
  
foo((value) => {  
    // runs after "some functionality"  
})
```

## Promise

Promises können klarer sein als Callbacks, sind aber auch weniger explizit und potenziell verwirrend.

```
const p1 = new Promise((resolve, reject) => {
    // some functionality

    resolve("Success!");
});
p1.then((value) => {
    // runs after "some functionality"
})
;
```

# Single-Page-Applikationen implementieren

# Progressive Web Apps

# Local First

1. No spinners: your work at your fingertips
2. Your work is not trapped on one device
3. The network is optional
4. Seamless collaboration with your colleagues
5. The Long Now
6. Security and privacy by default
7. You retain ultimate ownership and control

<https://www.inkandswitch.com/local-first/>

<https://localfirstweb.dev/>

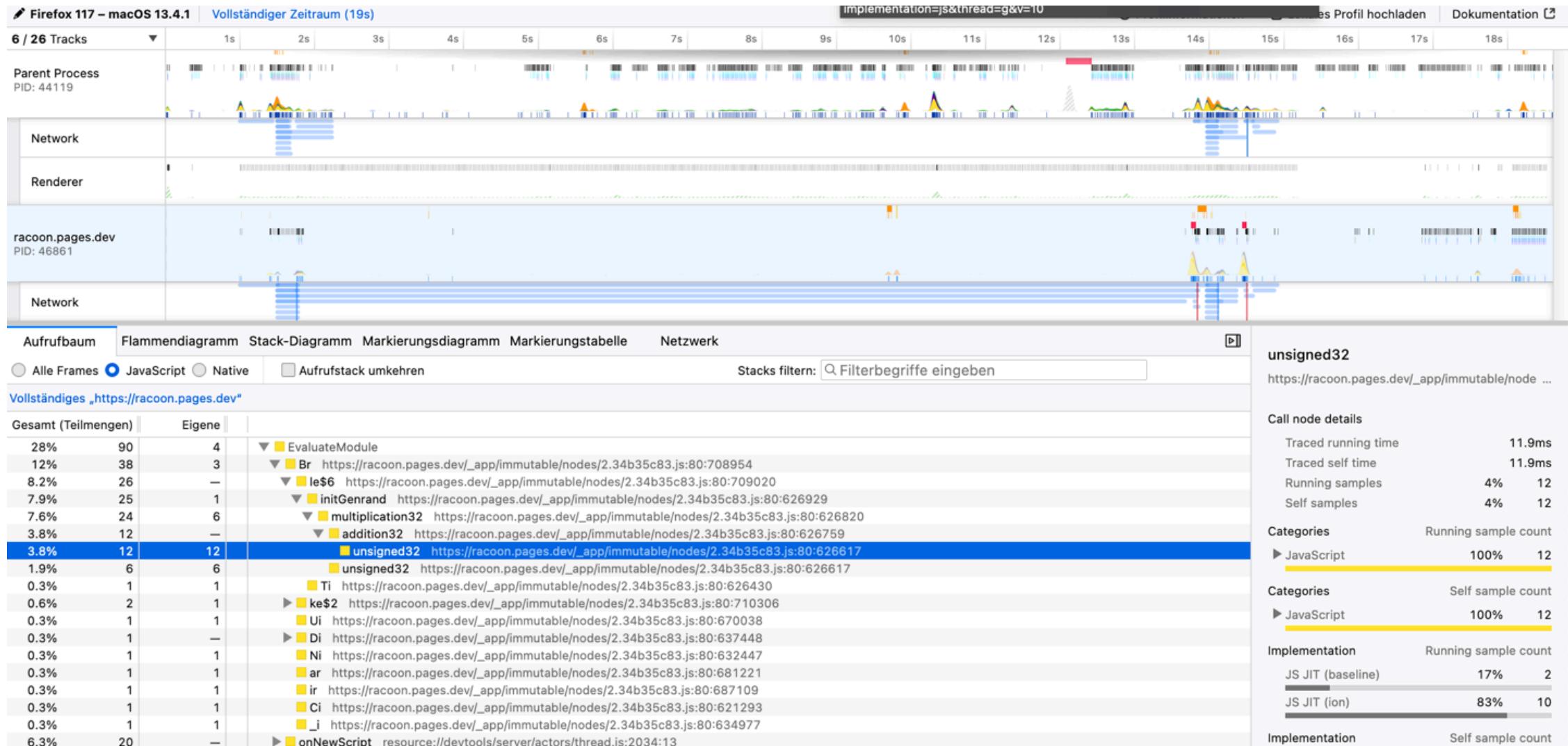
# Die Performance von Webanwendungen optimieren

# Browser Tools: Netzwerkanalyse

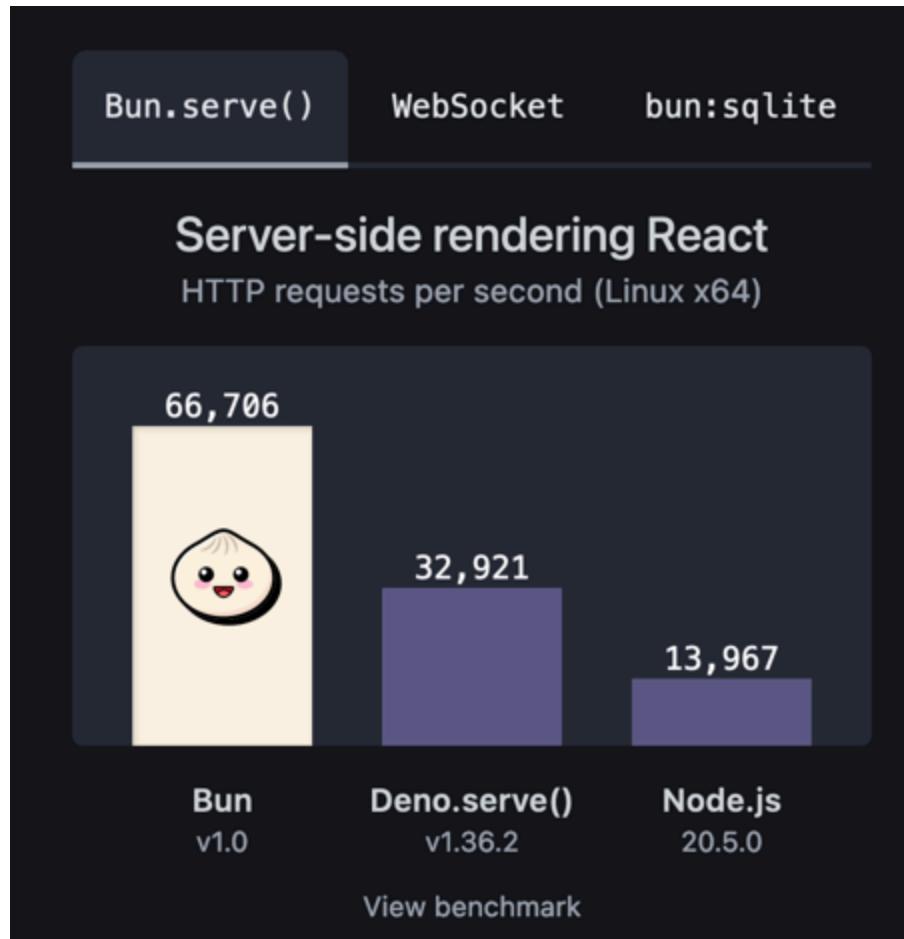
The screenshot shows the Network tab of a browser's developer tools. A red circle highlights the 'Cache deaktivieren' (Disable Cache) checkbox at the top right of the table header. The table lists network requests with the following data:

Status	Methode	Host	Datei	0 ms	220 ms	640 ms
200	GET	raccoon.pages.dev	/		225 ms	
200	GET	raccoon.pages.dev	0.00e663c4.css		195 ms	
200	GET	raccoon.pages.dev	2.b6295d3e.css		194 ms	
	GET	raccoon.pages.dev	start.e3ca0381.js			
200	GET	raccoon.pages.dev	scheduler.e254920f.js		192 ms	
200	GET	raccoon.pages.dev	singletons.b431a2d1.js		239 ms	
200	GET	raccoon.pages.dev	index.85055f88.js		239 ms	
200	GET	raccoon.pages.dev	app.18cd8190.js		238 ms	
200	GET	raccoon.pages.dev	index.f59f0ba0.js		238 ms	
200	GET	raccoon.pages.dev	0.d922621b.js		238 ms	
	GET	raccoon.pages.dev	2.34b35c83.js			
	GET	raccoon.pages.dev	bern.png			
	GET	raccoon.pages.dev	raccoon.svg			

# Browser Tools: Laufzeitanalyse



# Bun



```
$ bun install
```

# Bun is an npm-compatible package manager.

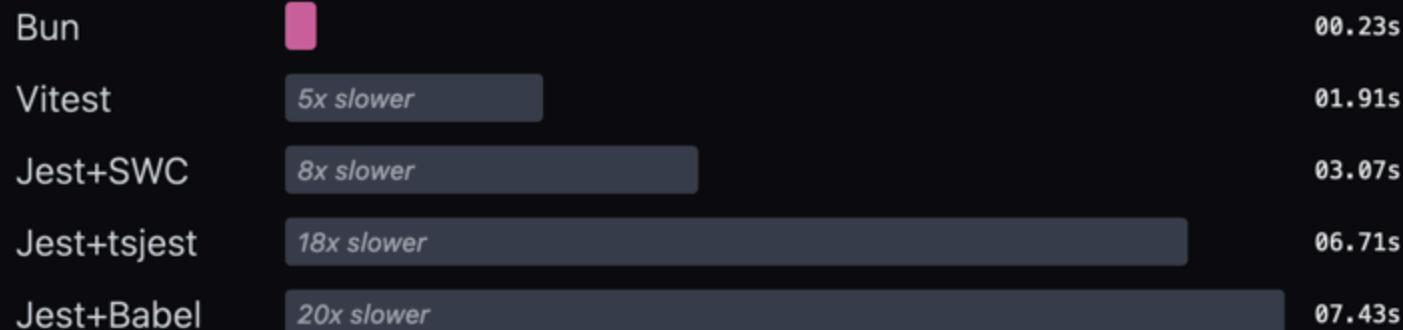
Bun		00.36s
pnpm	<i>17x slower</i>	06.44s
npm	<i>29x slower</i>	10.58s
Yarn	<i>33x slower</i>	12.08s

Installing dependencies from cache for a Remix app.

[View benchmark](#)

```
$ bun test
```

**Bun is a test runner that makes the rest look like test walkers.**



<https://bun.sh/>

# Energy, Time, Memory Comparision

Table 4: Normalized global results for Energy, Time, and Memory

Total			
	Energy (J)	Time (ms)	Mb
(c) C	1.00	(c) C	1.00
(c) Rust	1.03	(c) Rust	1.04
(c) C++	1.34	(c) C++	1.56
(c) Ada	1.70	(c) Ada	1.85
(v) Java	1.98	(v) Java	1.89
(c) Pascal	2.14	(c) Chapel	2.14
(c) Chapel	2.18	(c) Go	2.83
(v) Lisp	2.27	(c) Pascal	3.02
(c) Ocaml	2.40	(c) Ocaml	3.09
(c) Fortran	2.52	(v) C#	3.14
(c) Swift	2.79	(v) Lisp	3.40
(c) Haskell	3.10	(c) Haskell	3.55
(v) C#	3.14	(c) Swift	4.20
(c) Go	3.23	(c) Fortran	4.20
(i) Dart	3.83	(v) F#	6.30
(v) F#	4.13	(i) JavaScript	6.52
(i) JavaScript	4.45	(i) Dart	6.67
(v) Racket	7.91	(v) Racket	11.27
(i) TypeScript	21.50	(i) Hack	26.99
(i) Hack	24.02	(i) PHP	27.64
(i) PHP	29.30	(v) Erlang	36.71
(v) Erlang	42.23	(i) Jruby	43.44
(i) Lua	45.98	(i) TypeScript	46.20
(i) Jruby	46.54	(i) Ruby	59.34
(i) Ruby	69.91	(i) Perl	65.79
(i) Python	75.88	(i) Python	71.90
(i) Perl	79.58	(i) Lua	82.91