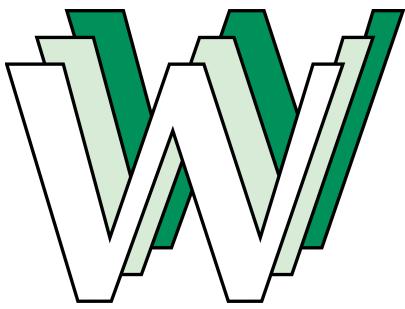


# **Grundlagen "Web-Engineering"**

# 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.

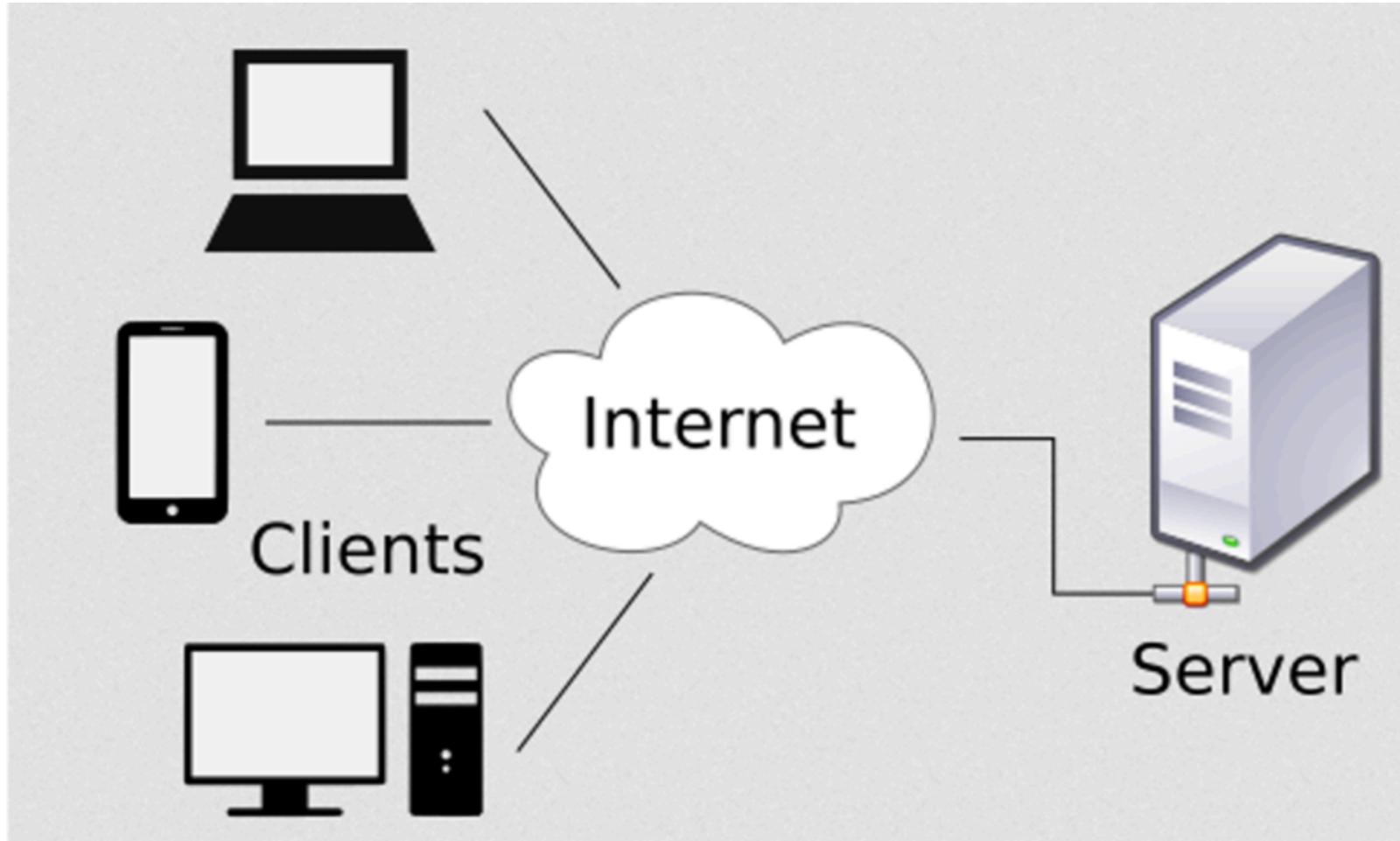
Die Webseiten enthalten meist **Texte**, oft mit **Bildern und grafischen Elementen** illustriert.  
Häufig sind auch **Videos, Tondokumente oder Musikstücke** eingebettet."

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))

# **Engineering**

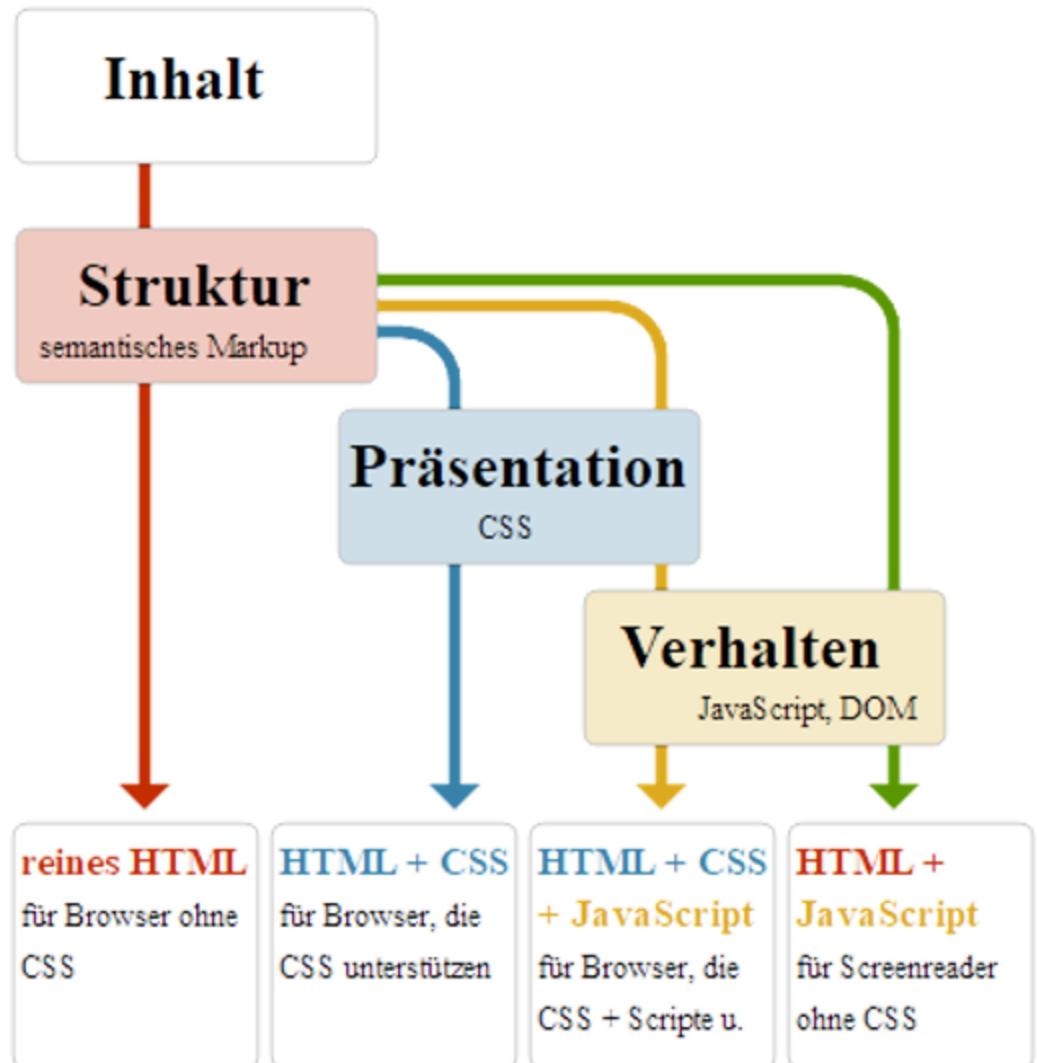
# Client - Server



## URL

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

# Aufbau von Webapplikationen



# **Software Stacks**

# LAMP

## *LAMP Stack*



Linux



Apache

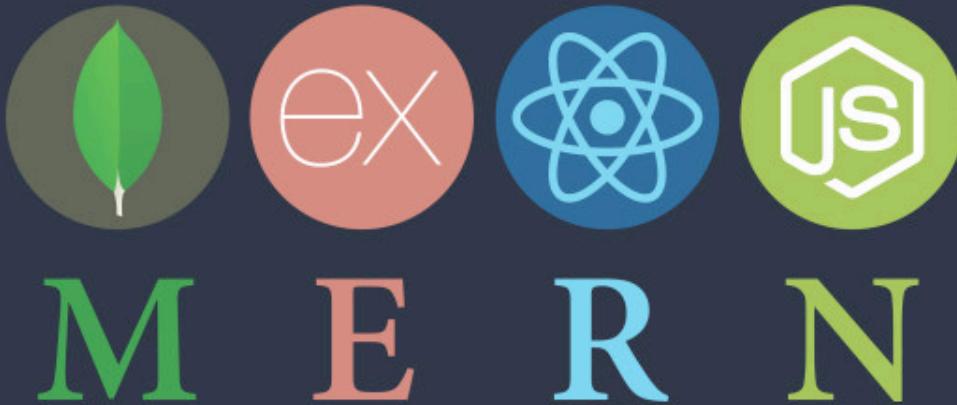


MySQL



PHP

# MERN



# Werkzeuge

Inspektor Konsole Debugger Netzwerkanalyse Stilbearbeitung Laufzeitanalyse Speicher ...

HTML durchsuchen + ⚒

Stile filtern :hov .cls + 📁

Element ⓘ { Inline

body, h1, dark.css:7 @prefers-color-scheme: dark  
h2, h3, h4,  
a, p, blockquote cite ⓘ {  
 color: #ddd;

body ⓘ { dark.css:2 @prefers-color-scheme: dark  
 background-color: #333;

body ⓘ { style.css:8  
 font-family: sans-serif;

body ⓘ { normalize.css:23  
 margin: 0;

Geerbt von html

html ⓘ { normalize.css:11  
 line-height: 1.15;  
 -webkit-text-size-adjust: 100%; ⚡

Layout Berechnet Änderungen Schriftarten

Flexbox

Flex-Behälter oder -Element auswählen, um fortzufahren.

Raster

Raster hervorheben

ul.wp-block-gallery.columns-2.is-cropped ⚡ ●

ul.wp-block-gallery.columns-3.is-cropped ⚡ ●

ul.wp-block-gallery.columns-3.is-cropped ⚡ ●

ul.wp-block-gallery.columns-3.is-cropped ⚡ ●

Anzeigeeinstellungen für Raster

Zeilennummern anzeigen

Bereichsnamen anzeigen

Unendliche Linien

Box-Modell

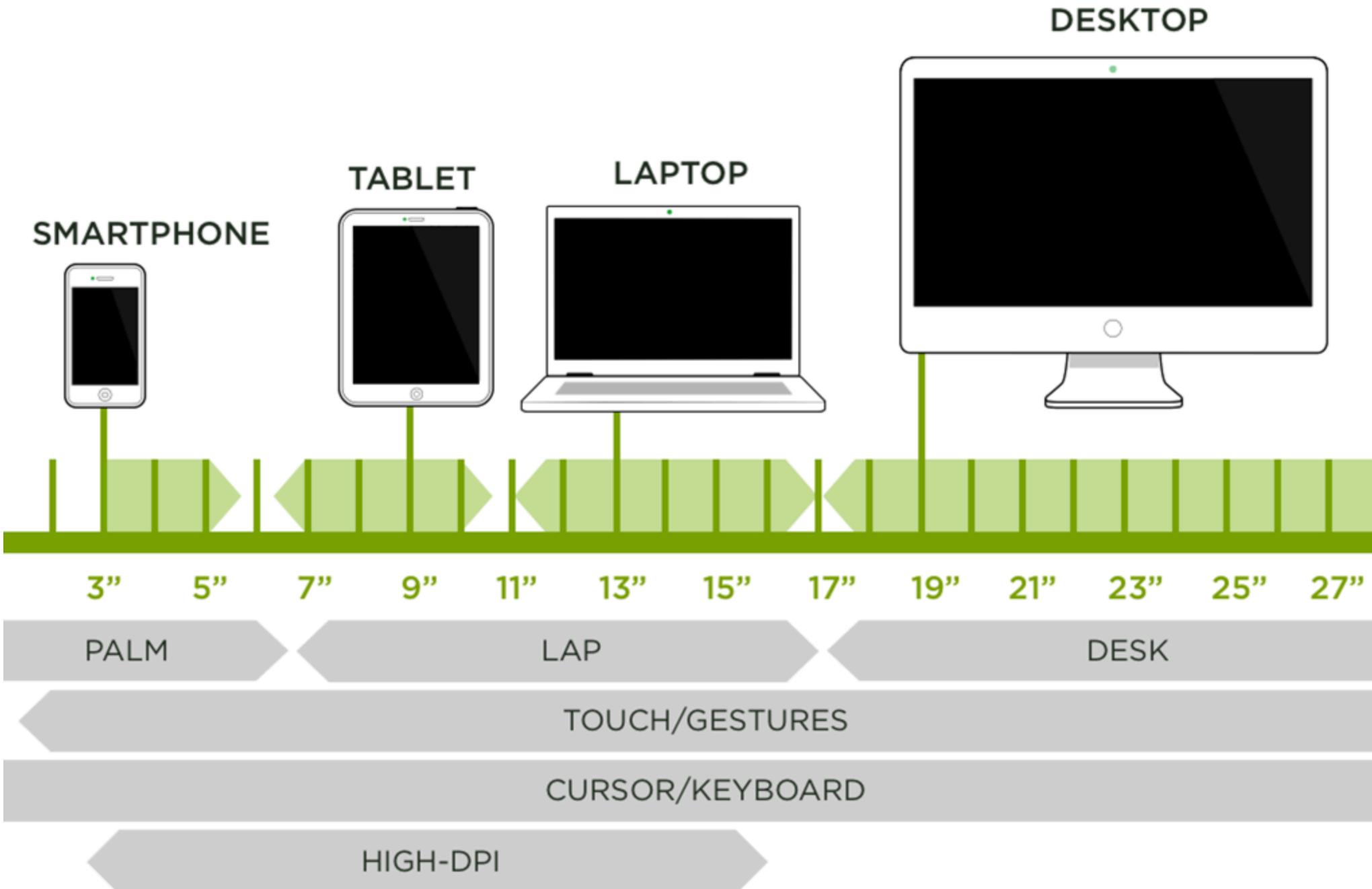
The diagram illustrates the Box Model for an element. It shows a light purple rectangular area representing the content, surrounded by a darker purple area for padding, a thin black border, and an outermost yellow dashed area for margin. The total width of the element is 823 and the height is 2397.2. The margin values are all 0, the padding values are all 0, the border thickness is 0, and the content area dimensions are 823x2397.2.

margin 0  
border 0  
padding 0  
823x2397.2 0 0 0  
0 0 0 0

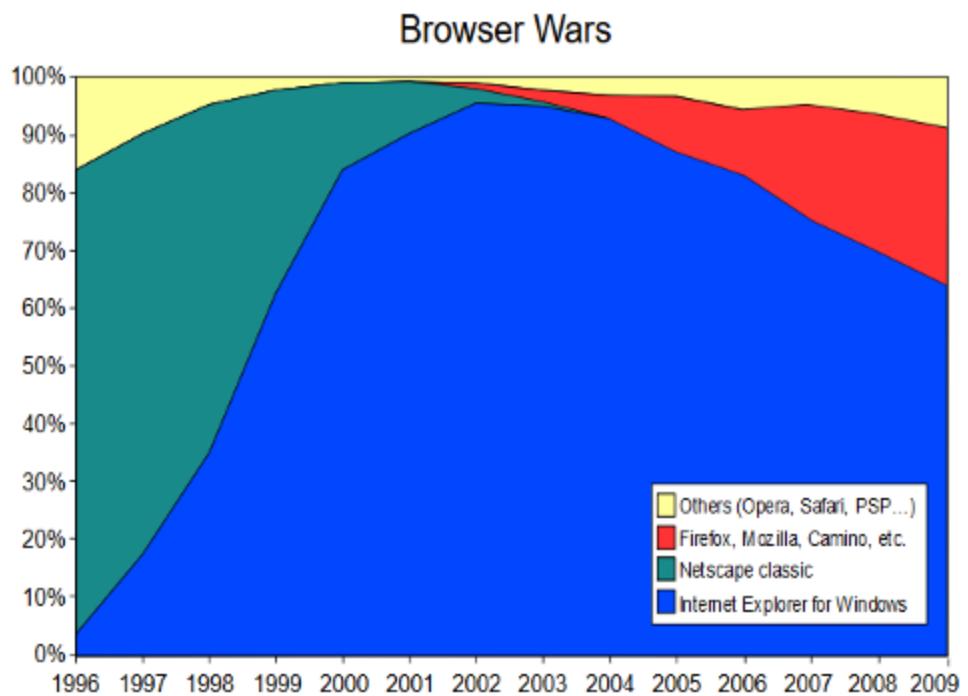
html > body

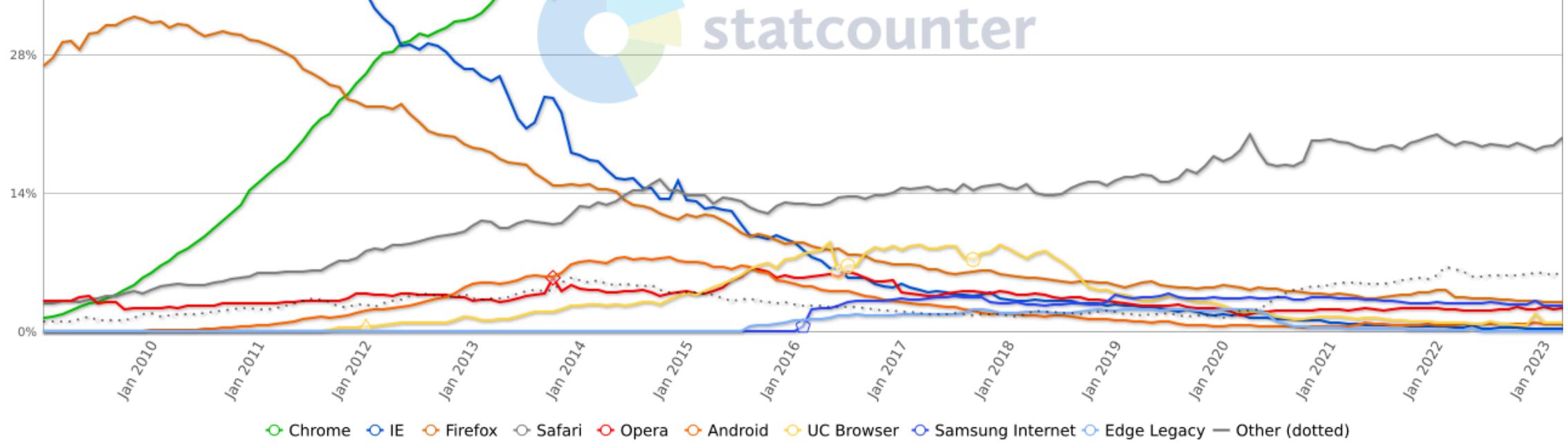
# **Webseiten strukturieren mit HTML**

# **Barrierefreiheit**

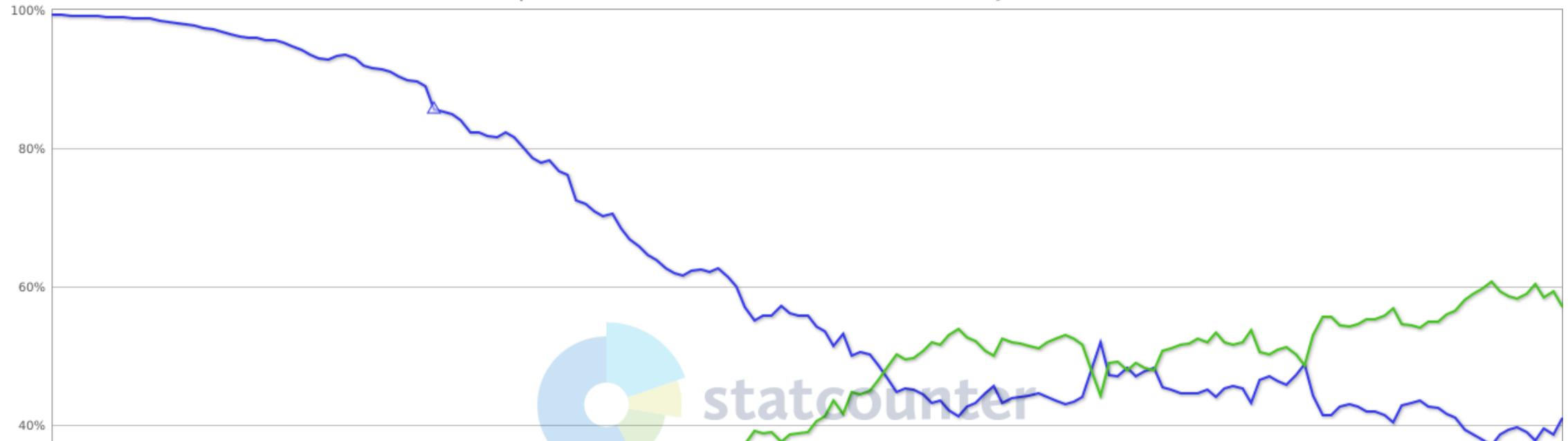


# **Browser Wars**





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



# 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

# Grundstruktur

```
<!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>
```

# **Webseiten gestalten mit CSS**

# Box Model

▼ Box Model

The diagram illustrates the standard box model with a central light blue box labeled "300x150". It is surrounded by a black border, a purple padding layer, and a yellow margin layer. The total width is 390px and height is 240px. The margin is 40px on all sides, the padding is 5px on all sides, and the border is 40px on all sides.

390x240 static

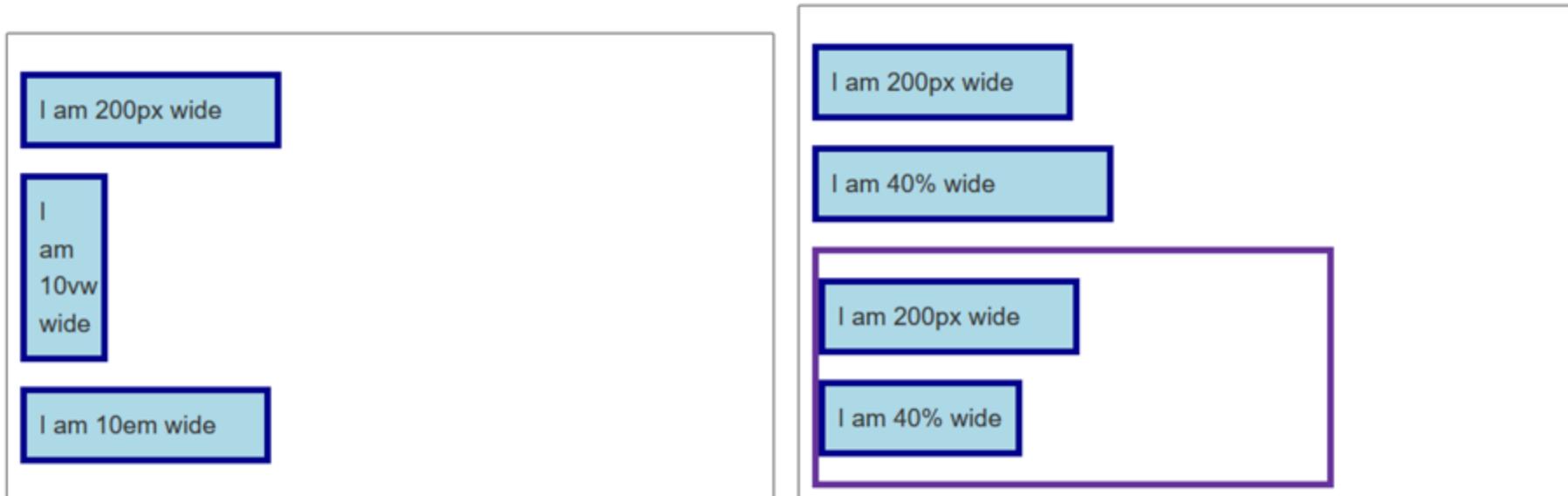
▼ Box Model Properties

box-sizing	content-box	line-height	28.8px
display	block	position	static
float	none	z-index	auto

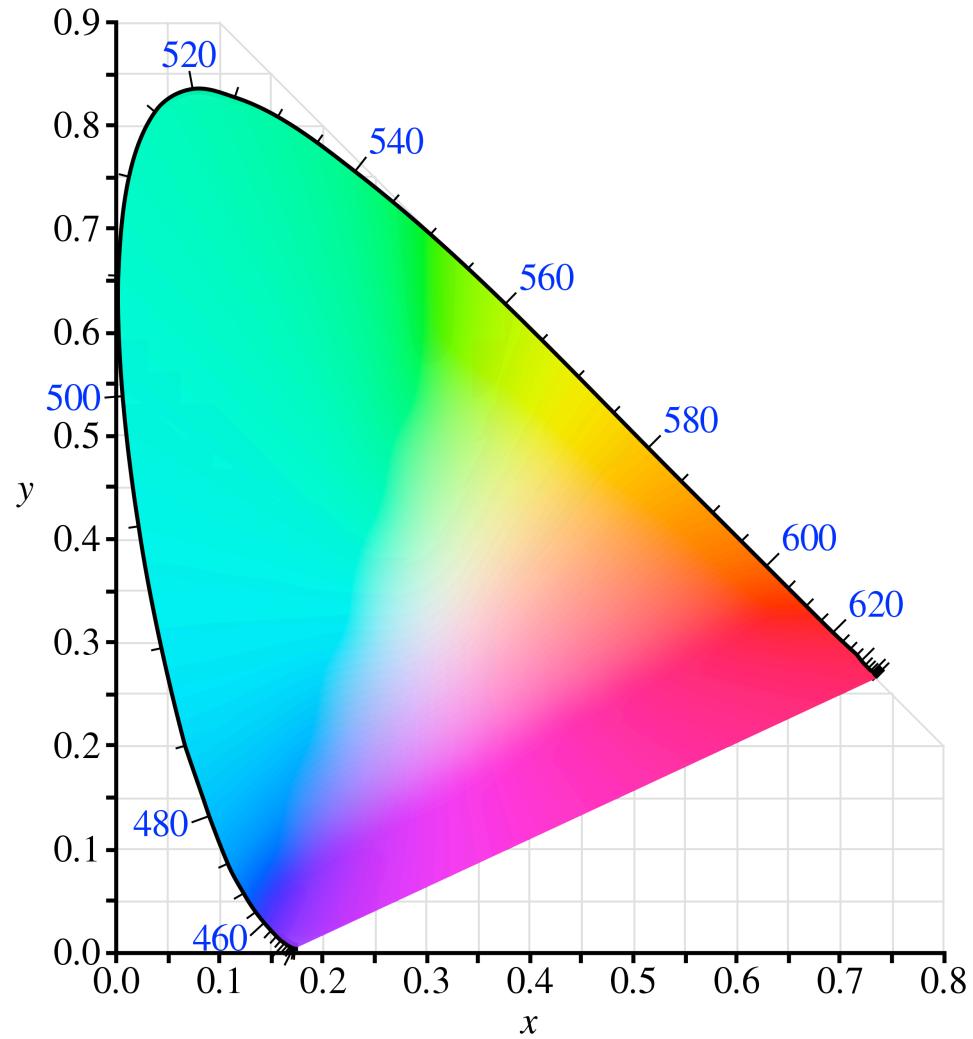
The diagram compares the standard box model (left) with the alternate box model (right). The standard box model shows a light blue content area with a purple border, a yellow padding area, and a white margin area. The alternate box model shows a light blue content area with a purple border, a white padding area, and a yellow margin area. A text box in the standard model says "use the standard box model." and a text box in the alternate model says "use the alternate box model."

# Einheiten

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



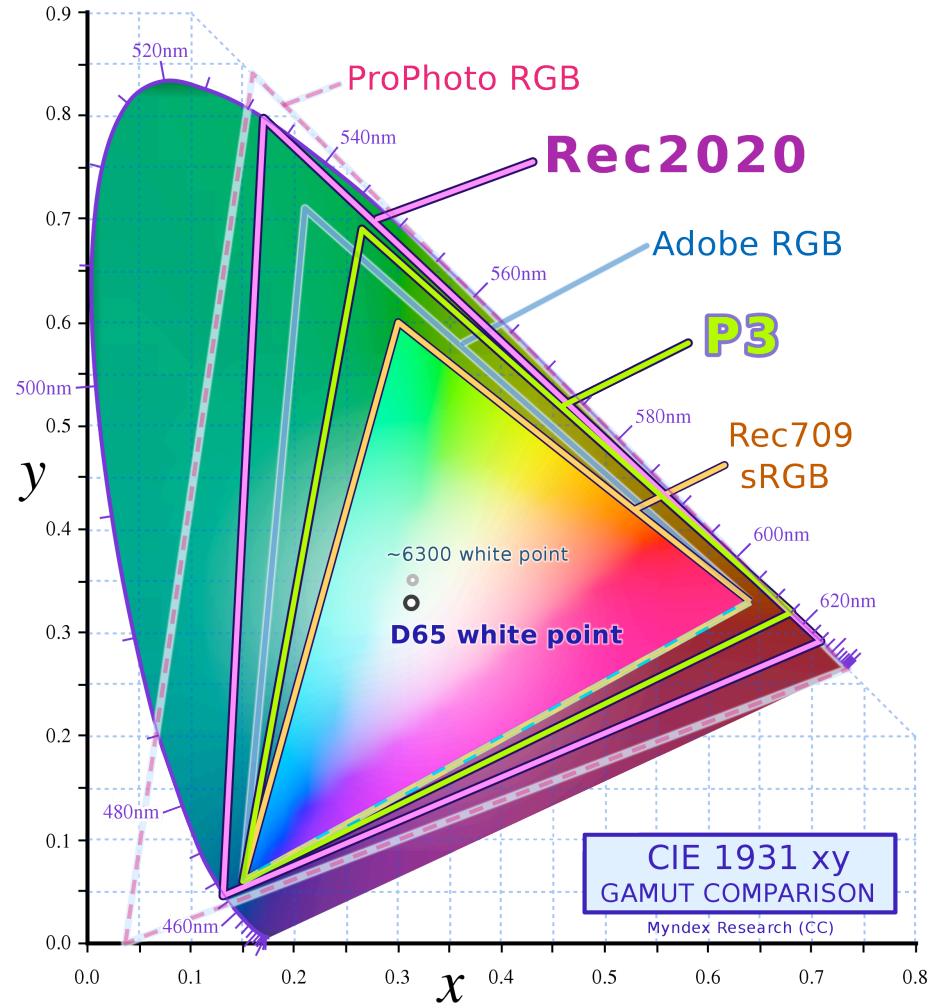
# Farben



CIE 1931 Farbraum



# Vergleich Farbräume





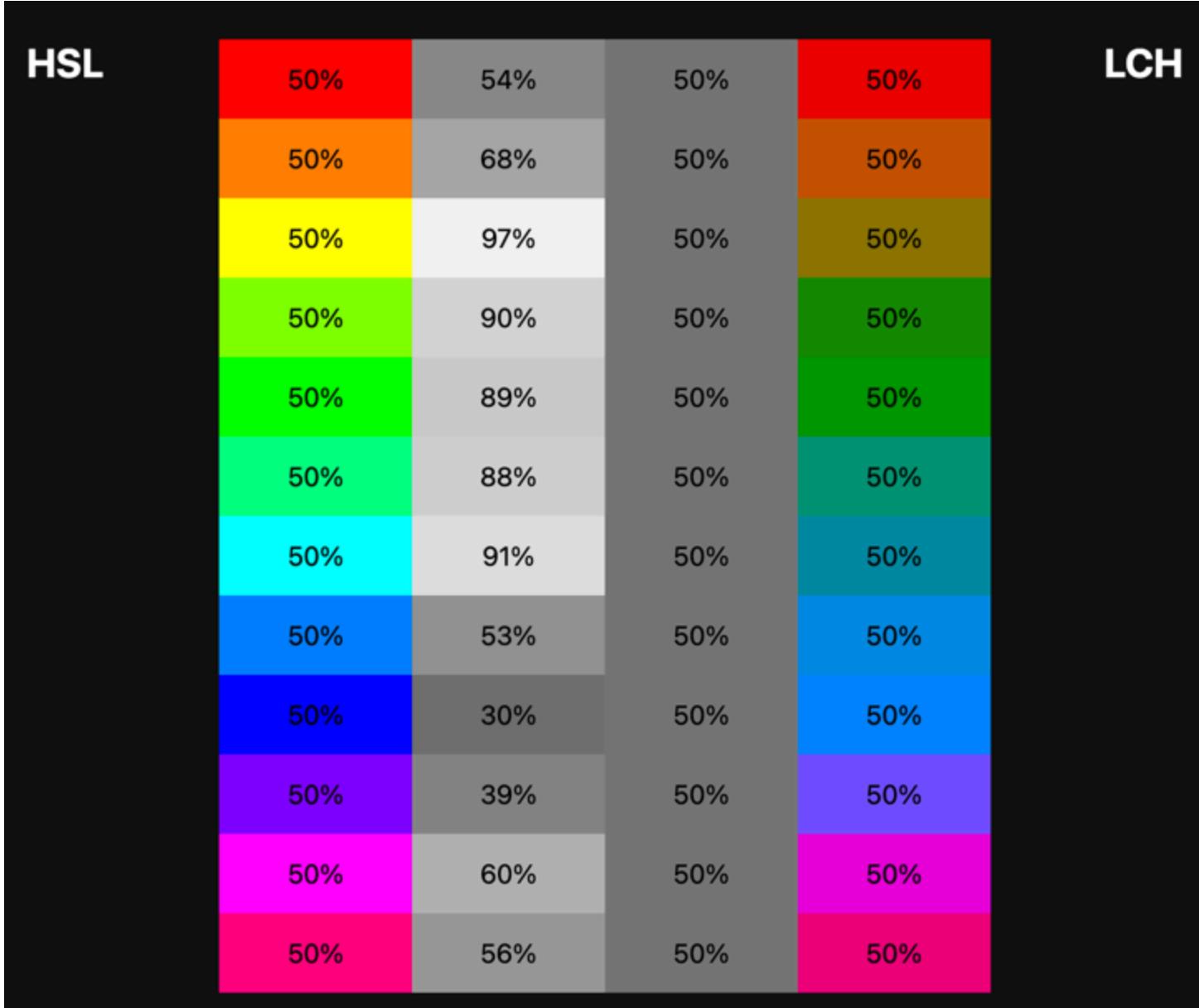
# Farben in CSS

## 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

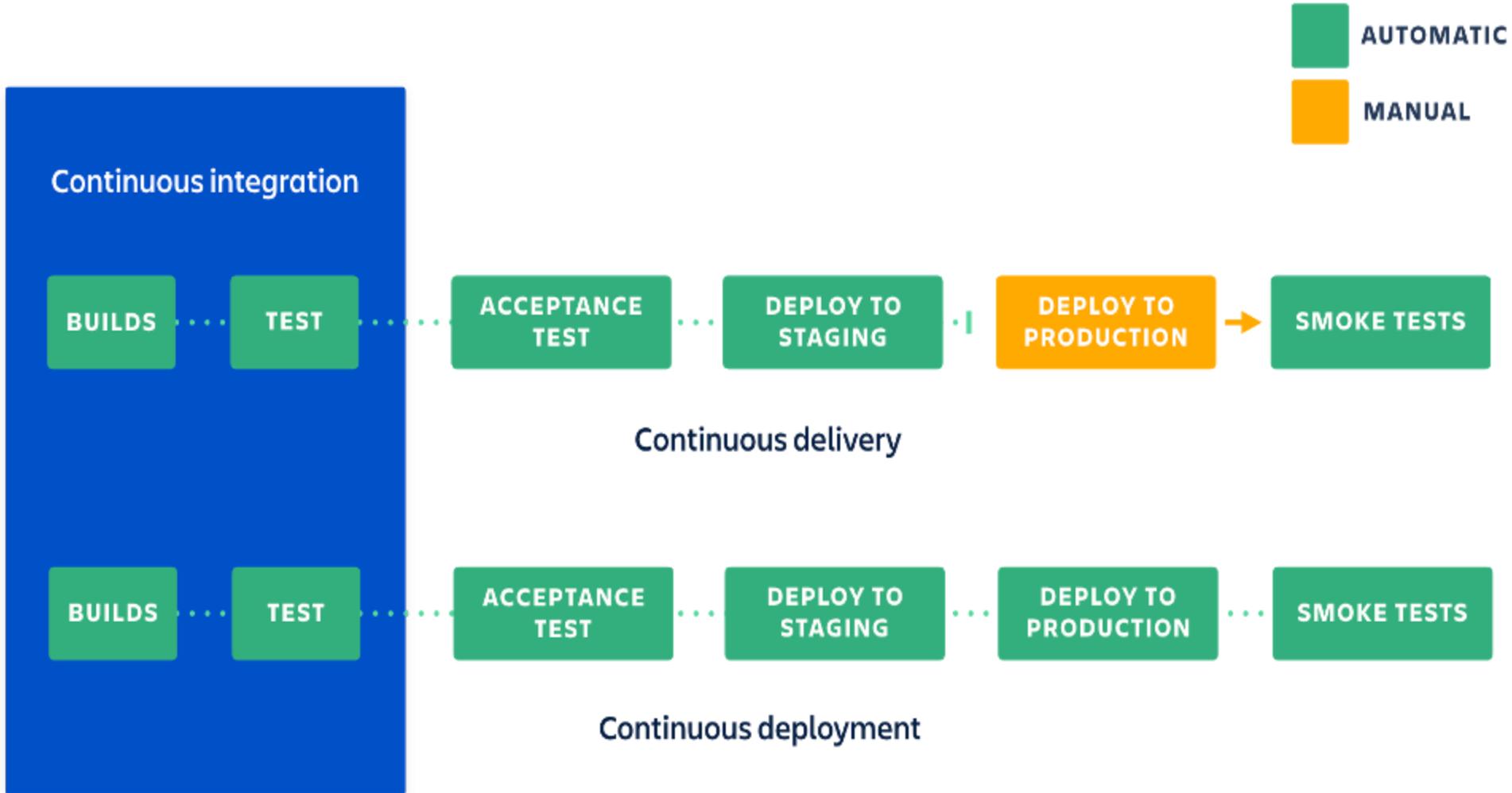
- LCH (Lightness Chroma Hue / Opacity): `color: lch(29.2345% 44.2 27 / 0.5)`
- Oklch: `color: oklch(40.1% 0.123 21.57)`
- CIELAB (Lightness, red-green, blue-yellow): `color: lab(29.2345% 39.3825 20.0664);`
- Oklab: `color: oklab(40.1% 0.1143 0.045);`



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

# Webanwendungen deployen und hosten

# CI / CD



# Continuous Integration

- Kein Branching, alle Änderungen werden von allen Teammitgliedern mehrmals täglich in den Master Branch eingeccheckt.
- Dieser Branch ist jederzeit lauffähig
- Dadurch werden die Releases vereinfachen
- Eine sehr hohe, automatische Testabdeckung ist zwingend

# Continuous Delivery

- Low risk releases
- Faster time to market
- Higher quality
- Lower costs
- Better products
- Happier teams

# Principles

- Build quality in
- Work in small batches
- Computers perform repetitive tasks, people solve problems
- Relentlessly pursue continuous improvement
- Everyone is responsible

<https://www.continuousdelivery.com/>

Modern Software Engineering

## Begriffe

GitOps: Git als Single Source of Truth für alles was für die Applikation relevant ist.

# **Webanwendungen organisieren und verwalten**

# **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.

## Komposition

- 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.

## 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.

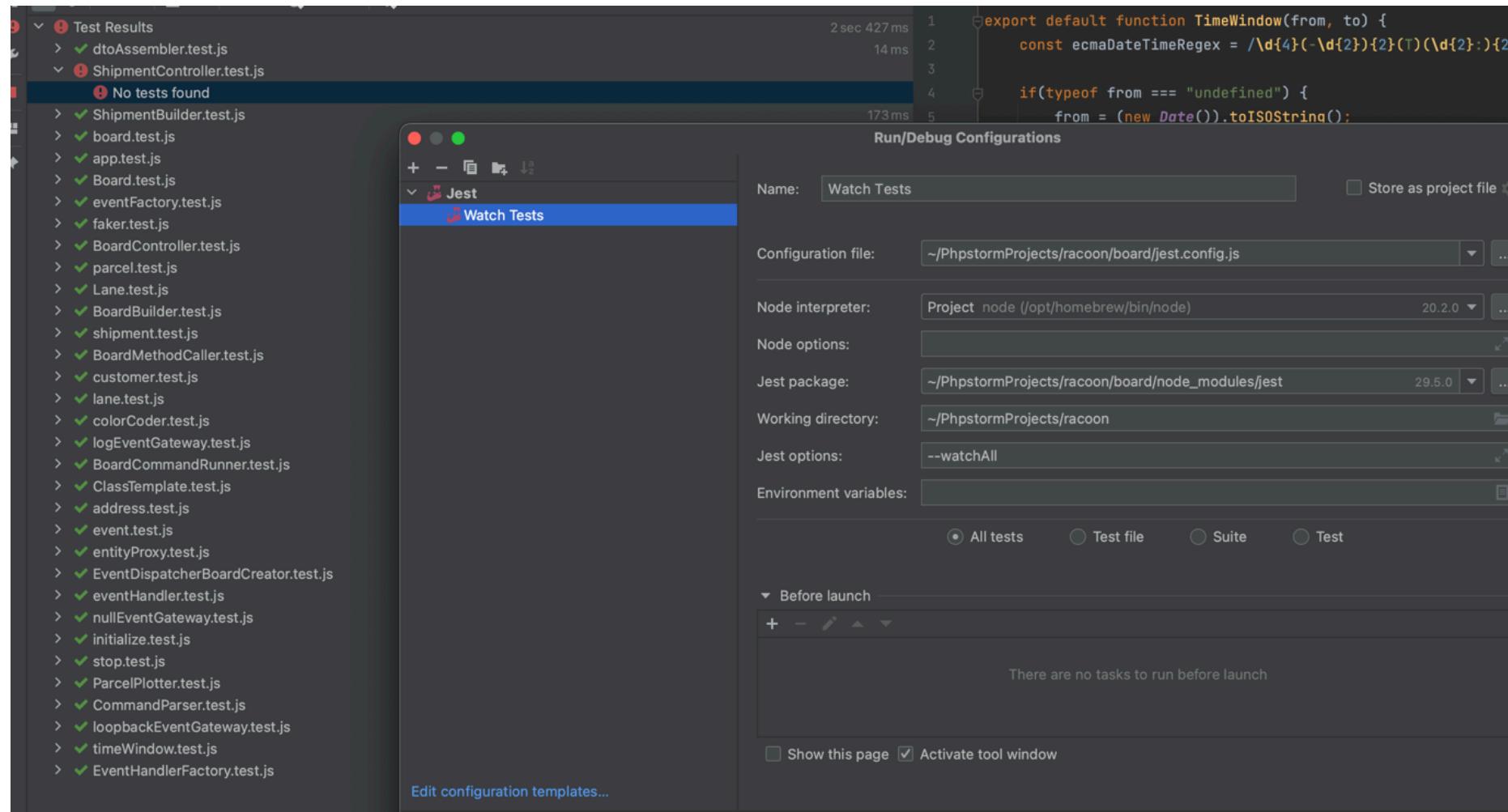
```
const p1 = new Promise((resolve, reject) => {
    // some functionality

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

# Webanwendungen testen

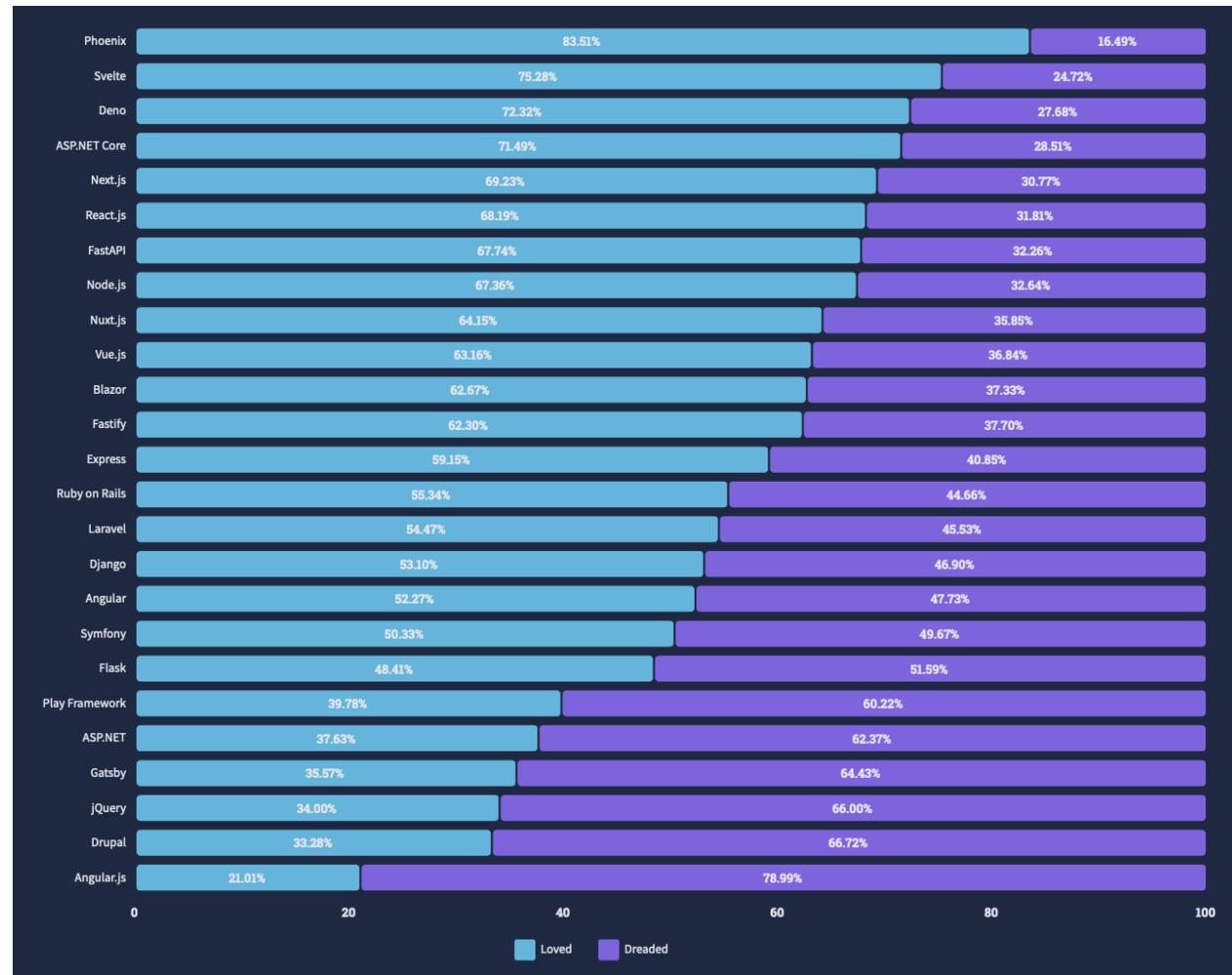
- Test First: Fokus auf die Problemstellung und Schnittstelle
- Nur eigenen Code testen. Datenbanken, APIs oder Libraries werden nur im Rahmen von Integrationstests aufgerufen.
- Tests geben eine Rückmeldung zum Code: Wenn Code schwierig zu testen ist, sollte er vermutlich anders strukturiert werden.
- **Humble Object**: Code, der schwierig zu testen ist in einem minimalen Objekt isolieren

# IDE Integration



# Single-Page-Applikationen implementieren

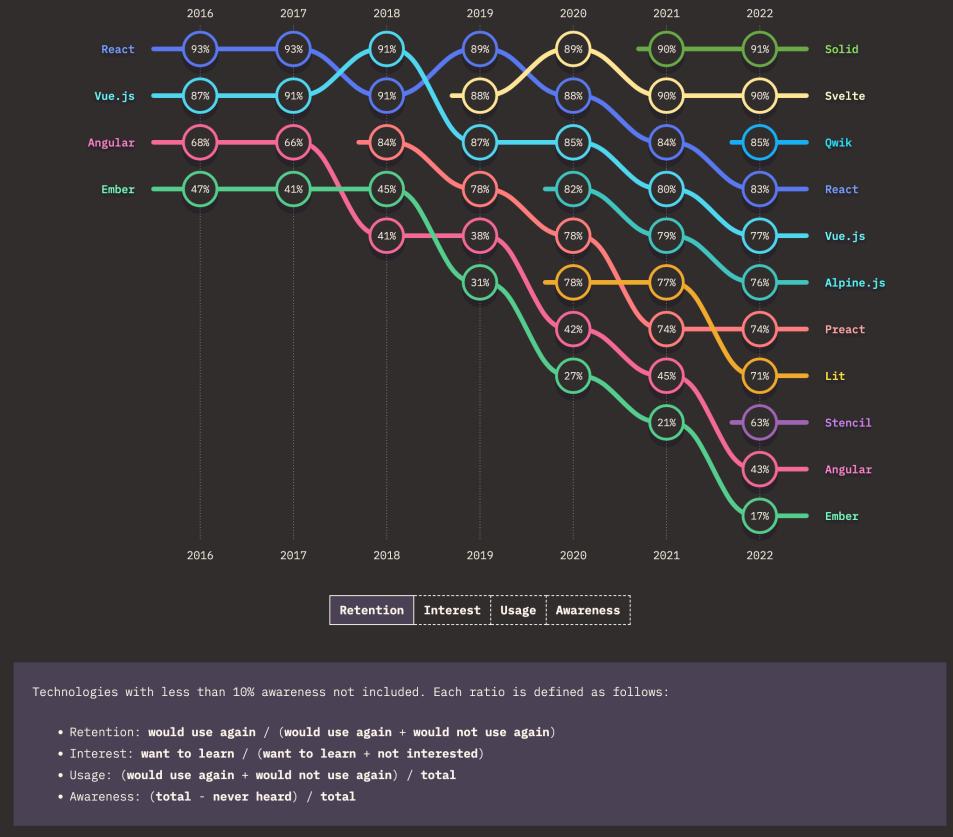
# Web Frameworks



Stackoverflow Survey 2022,  
21.06.2023

## RANKINGS OVER TIME

Retention, interest, usage, and awareness ratio rankings.



State of JavaScript 2022, 21.06.2023

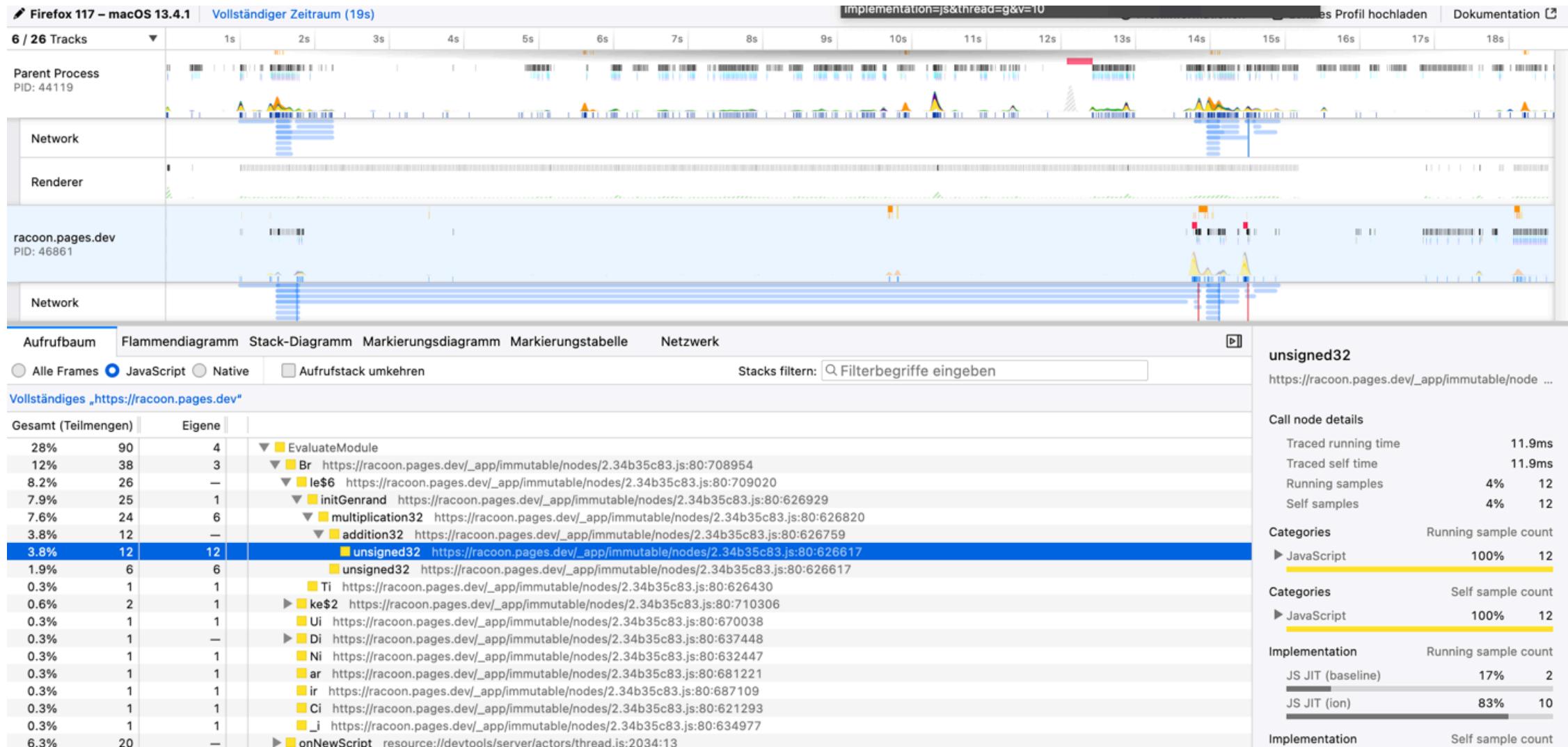
# Die Performance von Webanwendungen optimieren

# Browser Tools: Netzwerkanalyse

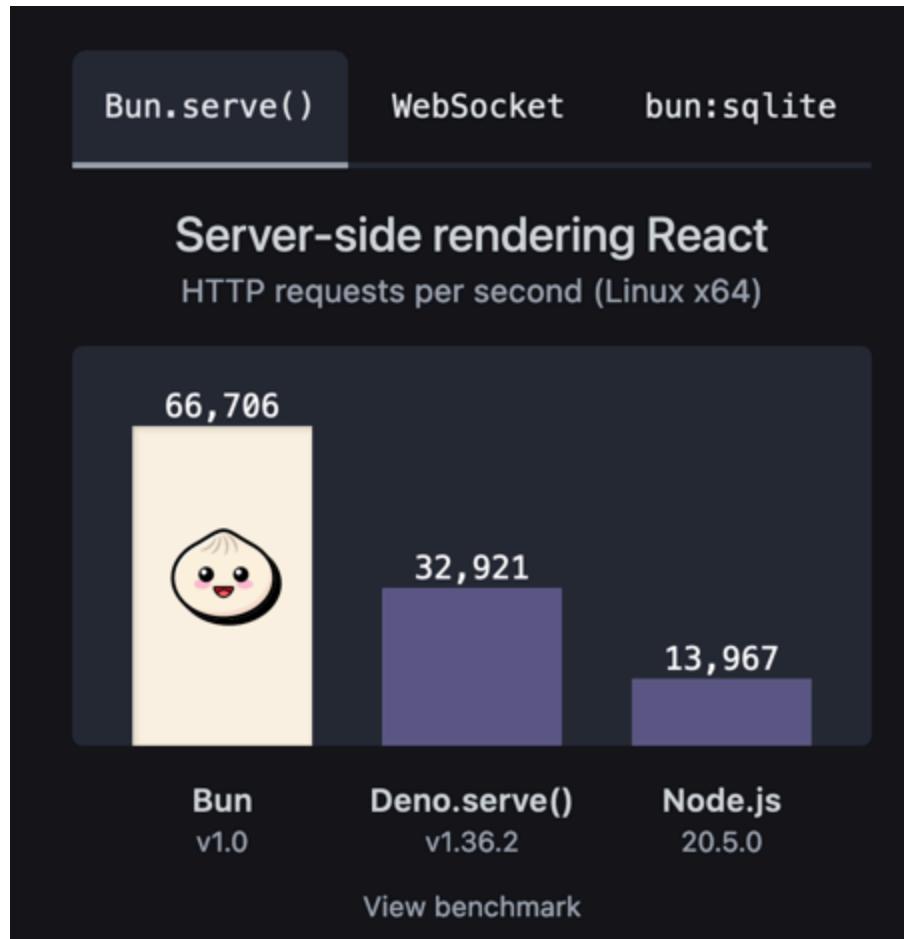
The screenshot shows the Network tab of a browser's developer tools. A red circle highlights the 'Cache deaktivieren' (Disable Cache) checkbox at the top right of the table header. The table lists network requests with the following data:

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.

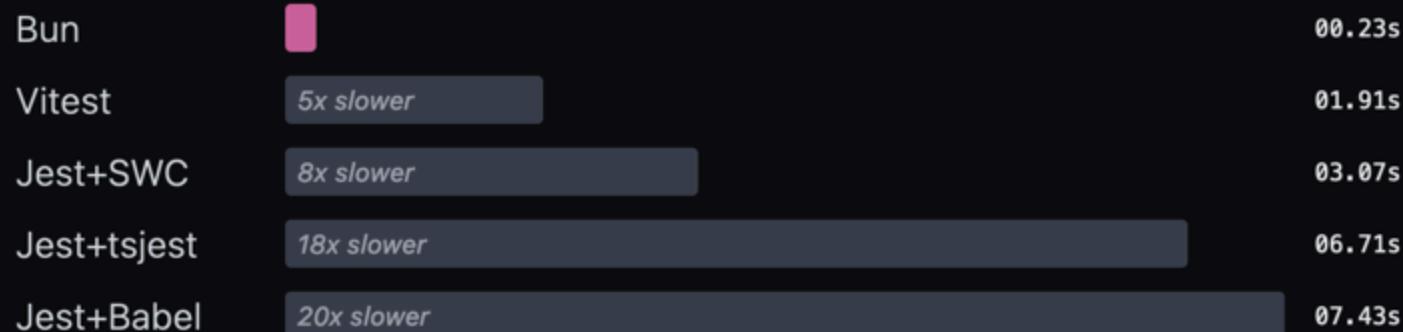
Bun		00.36s
pnpm	<i>17x slower</i>	06.44s
npm	<i>29x slower</i>	10.58s
Yarn	<i>33x slower</i>	12.08s

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/>

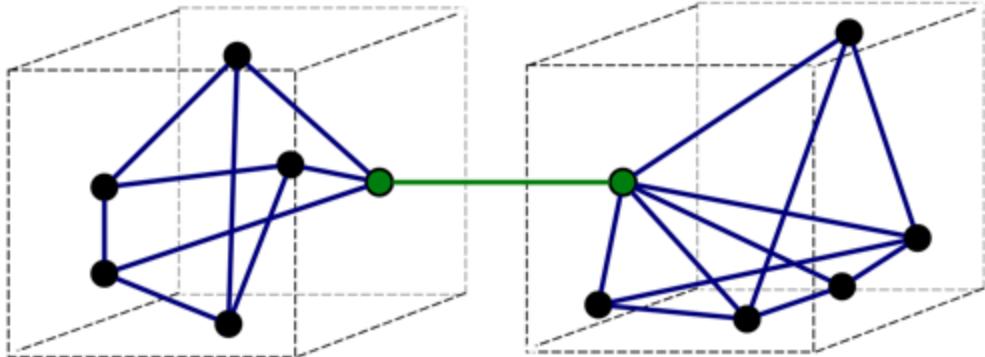
# Energy, Time, Memory Comparision

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

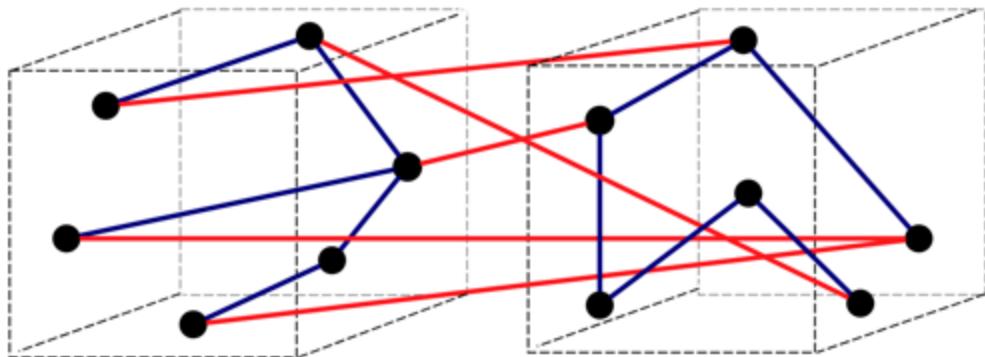
Total			
	Energy (J)	Time (ms)	Mb
(c) C	1.00	(c) C	1.00
(c) Rust	1.03	(c) Rust	1.04
(c) C++	1.34	(c) C++	1.56
(c) Ada	1.70	(c) Ada	1.85
(v) Java	1.98	(v) Java	1.89
(c) Pascal	2.14	(c) Chapel	2.14
(c) Chapel	2.18	(c) Go	2.83
(v) Lisp	2.27	(c) Pascal	3.02
(c) Ocaml	2.40	(c) Ocaml	3.09
(c) Fortran	2.52	(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

# **Webarchitekturen verstehen und einsetzen**

# High Cohesion - Low Coupling

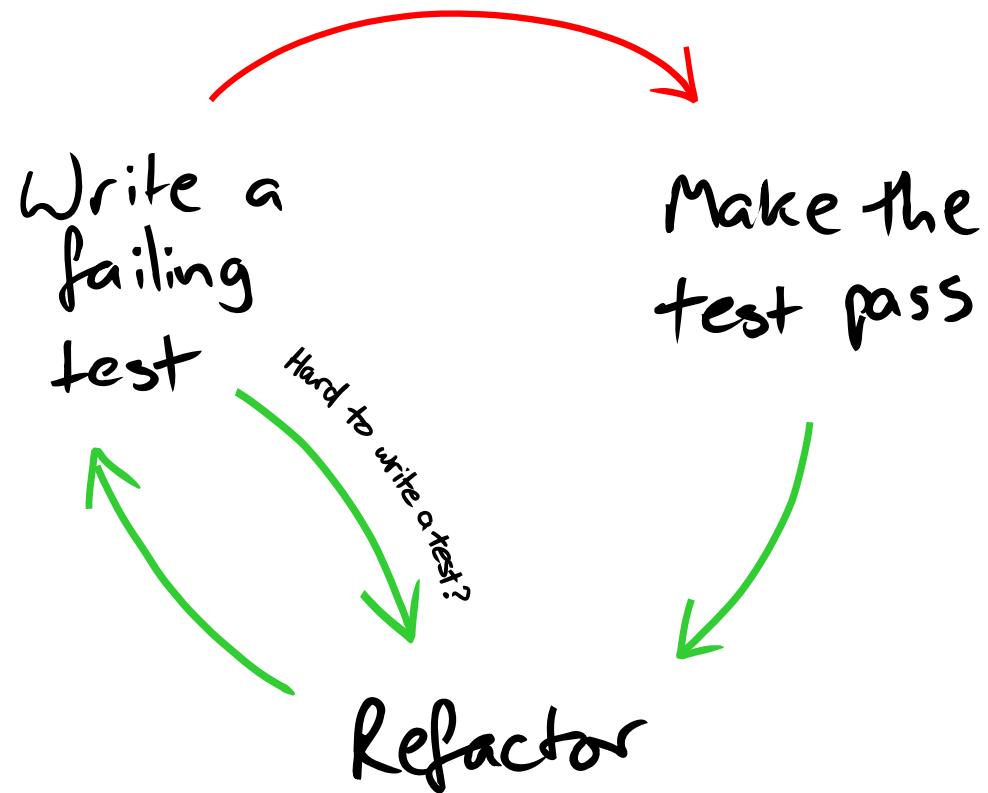


a) Good (loose coupling, high cohesion)

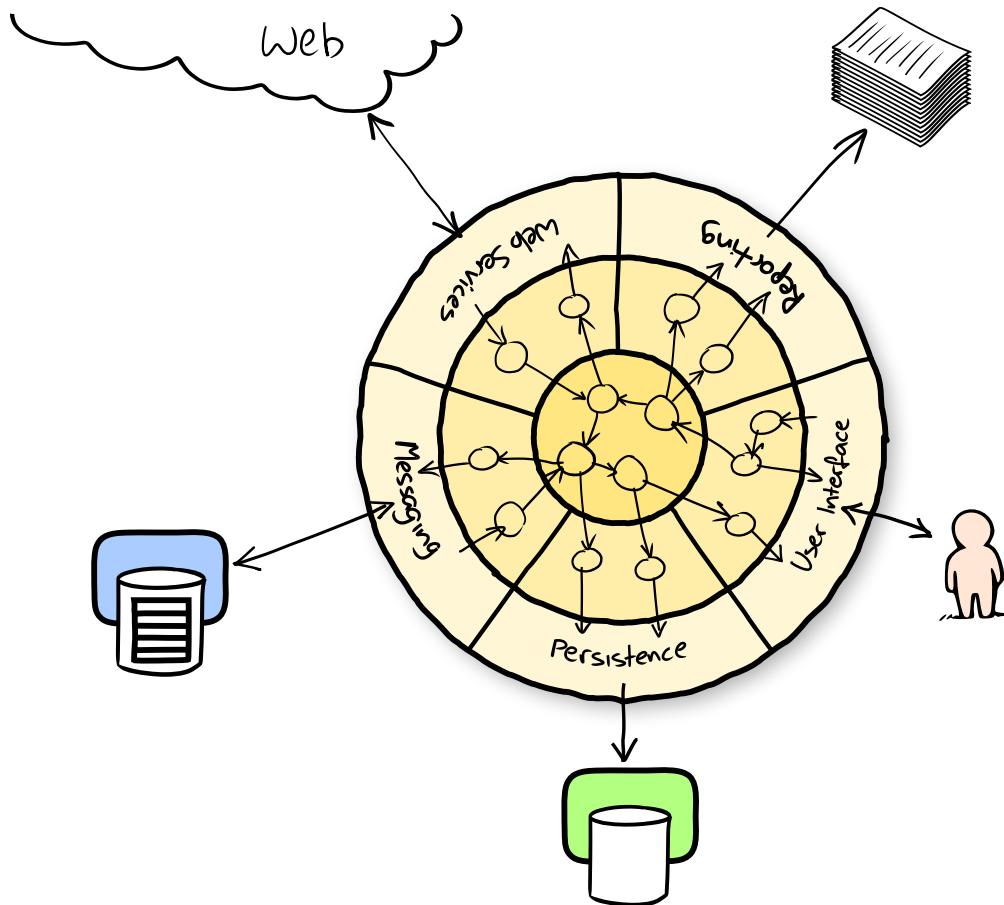


b) Bad (high coupling, low cohesion)

# Listening to Tests

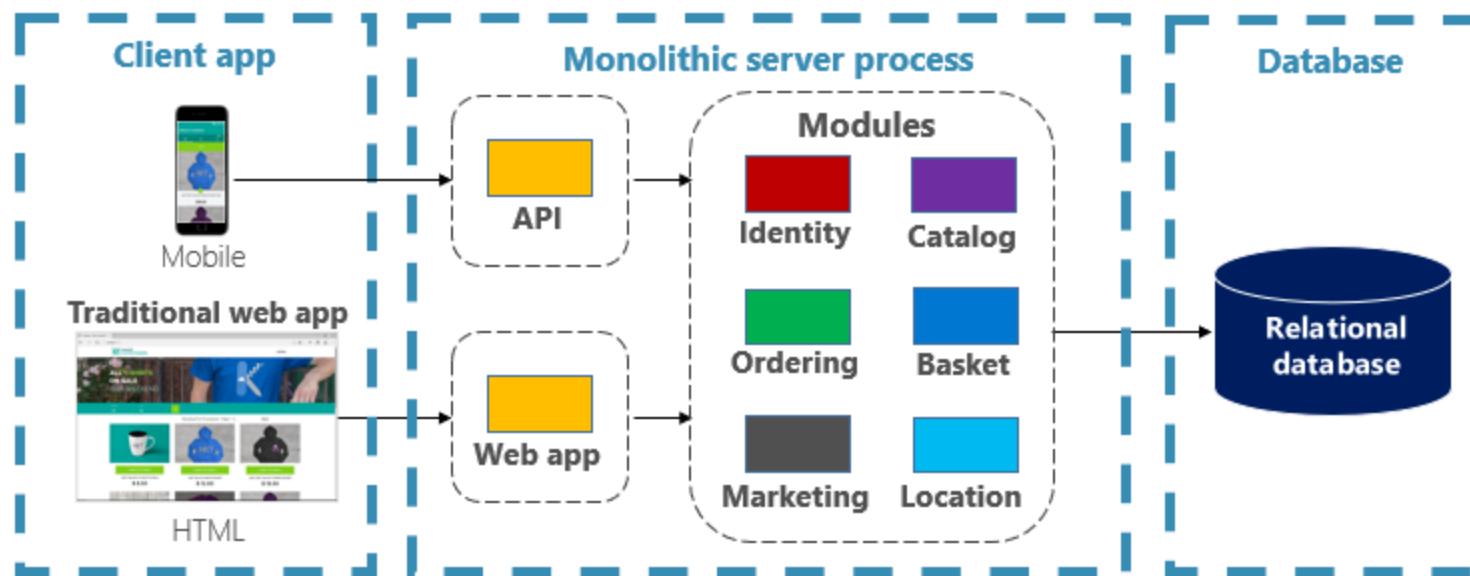


# Ports and Adapters

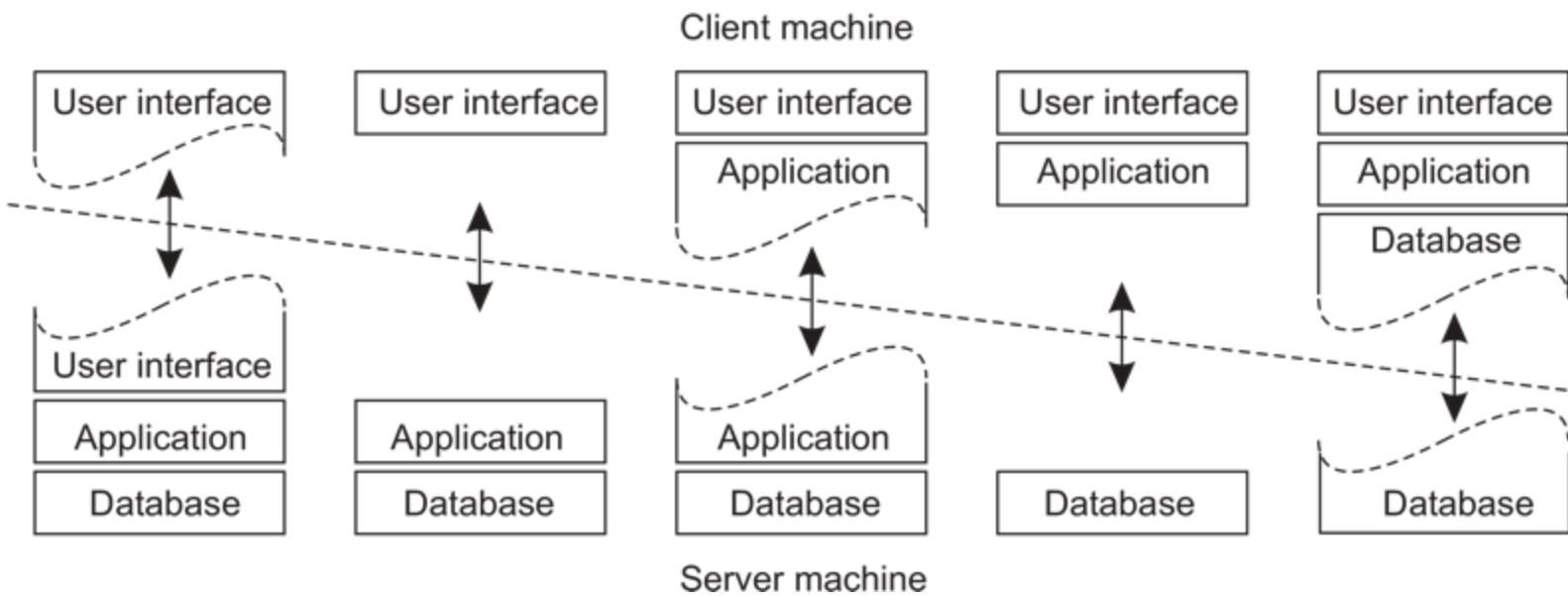


[growing-object-oriented-software.com](http://growing-object-oriented-software.com)

# Traditional Monolithic Design



# Schichtenarchitektur im Client Server Modell

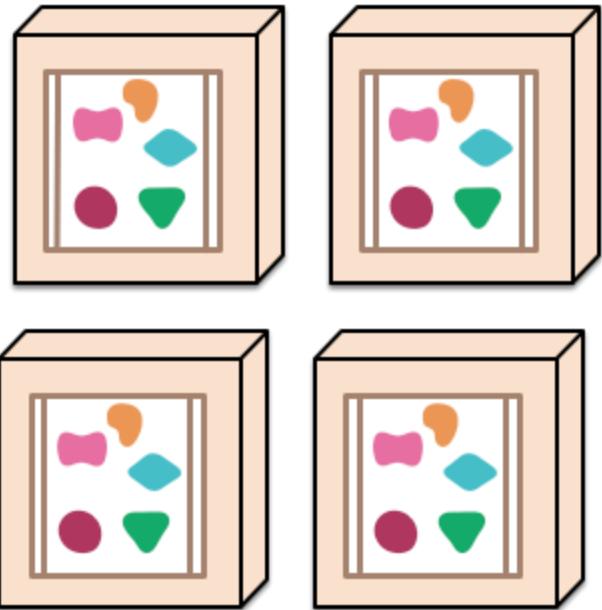


# Microservices

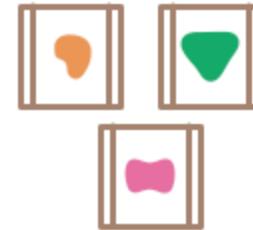
*A monolithic application puts all its functionality into a single process...*



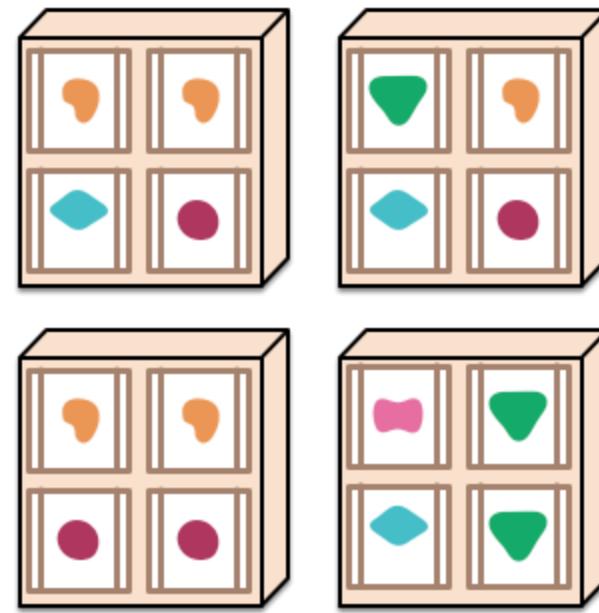
*... and scales by replicating the monolith on multiple servers*



*A microservices architecture puts each element of functionality into a separate service...*



*... and scales by distributing these services across servers, replicating as needed.*



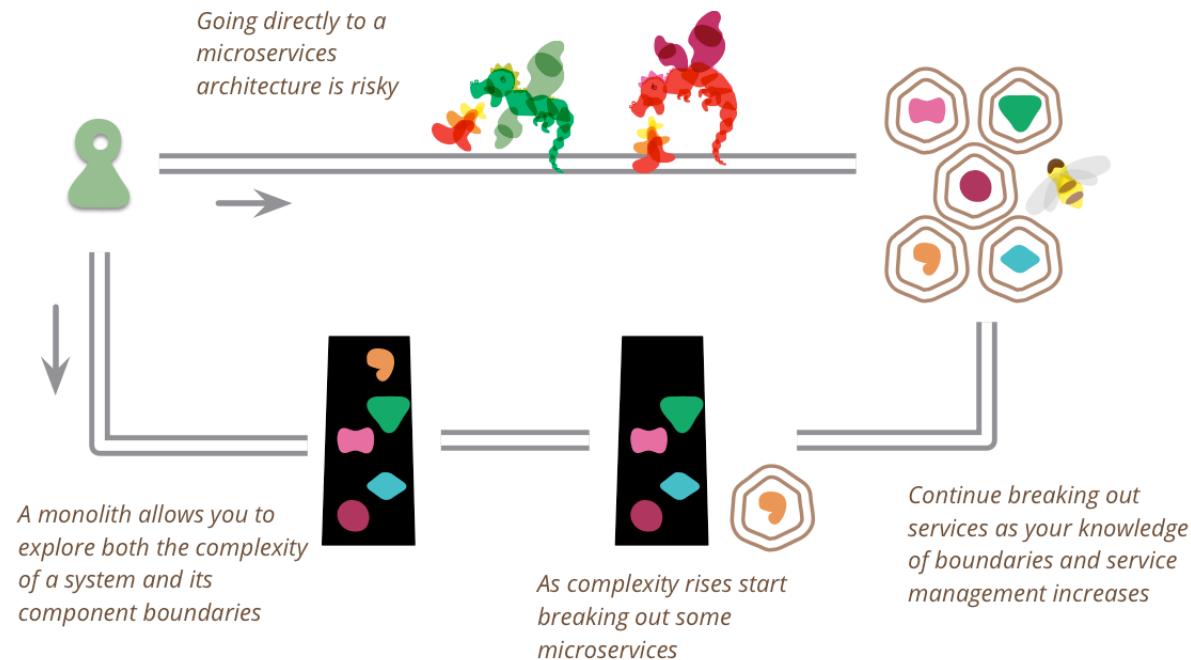
# Microservices

- Maximale Skalierbarkeit
- Einzelne Services können von **kleinen[^1]** Teams **unabhängig entwickelt und deployed** werden
- Bessere Wart- und Erweiterbarkeit
- Unterschiedliche Technologien können eingesetzt werden
- Kommunikation nicht trivial
- Höhere Wahrscheinlichkeit eines Ausfalls
- **Hohe Komplexität**

[^1]: "We try to create teams that are no larger than can be fed by two pizzas"

# Monolith First

- Vorsicht vor **Cargo-Kult**: Amazon, Google, Meta etc. haben heute andere Herausforderungen als Startups
- Technologien oder Architekturen wählen, "weil Google macht das auch so" ist ein schlechter Grund



[martinfowler.com/bliki/MonolithFirst.html](http://martinfowler.com/bliki/MonolithFirst.html)