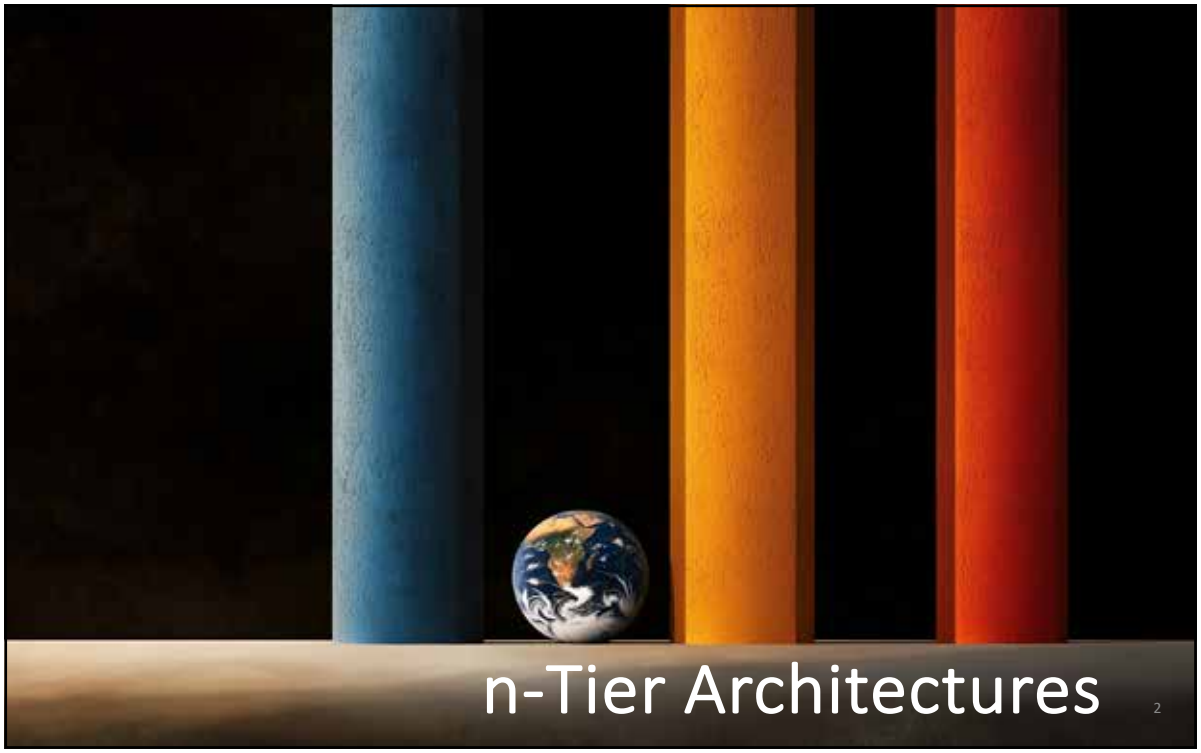
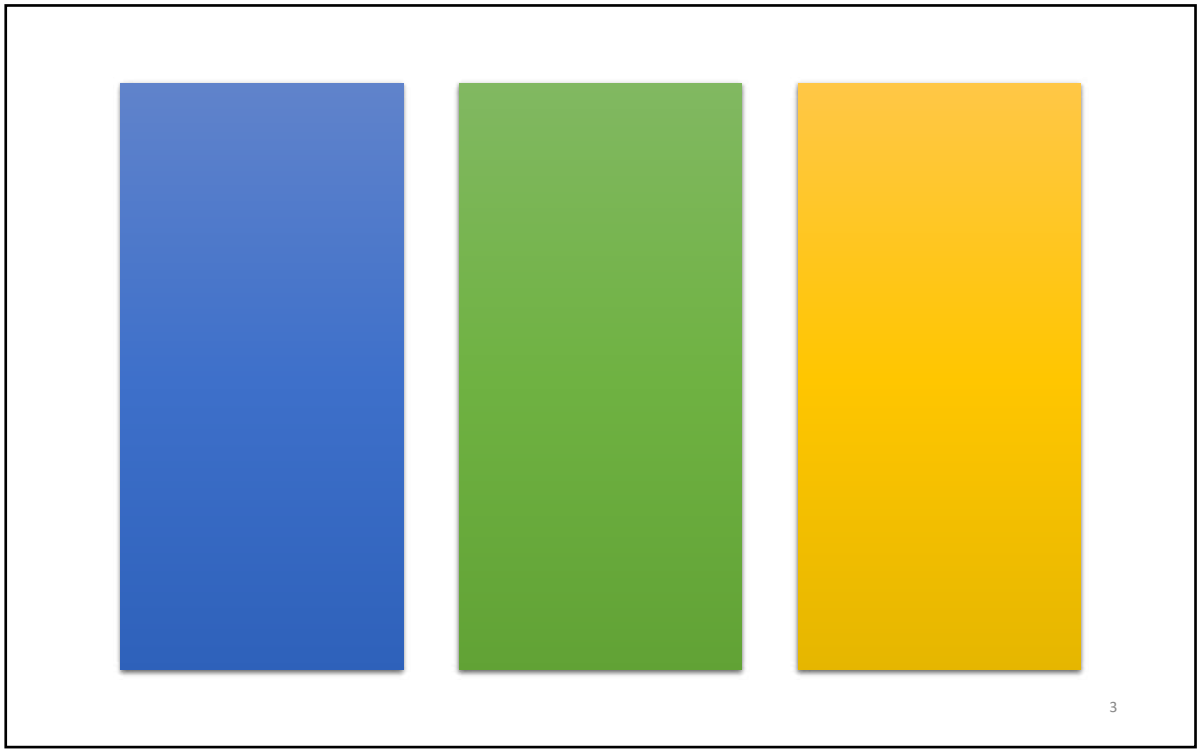
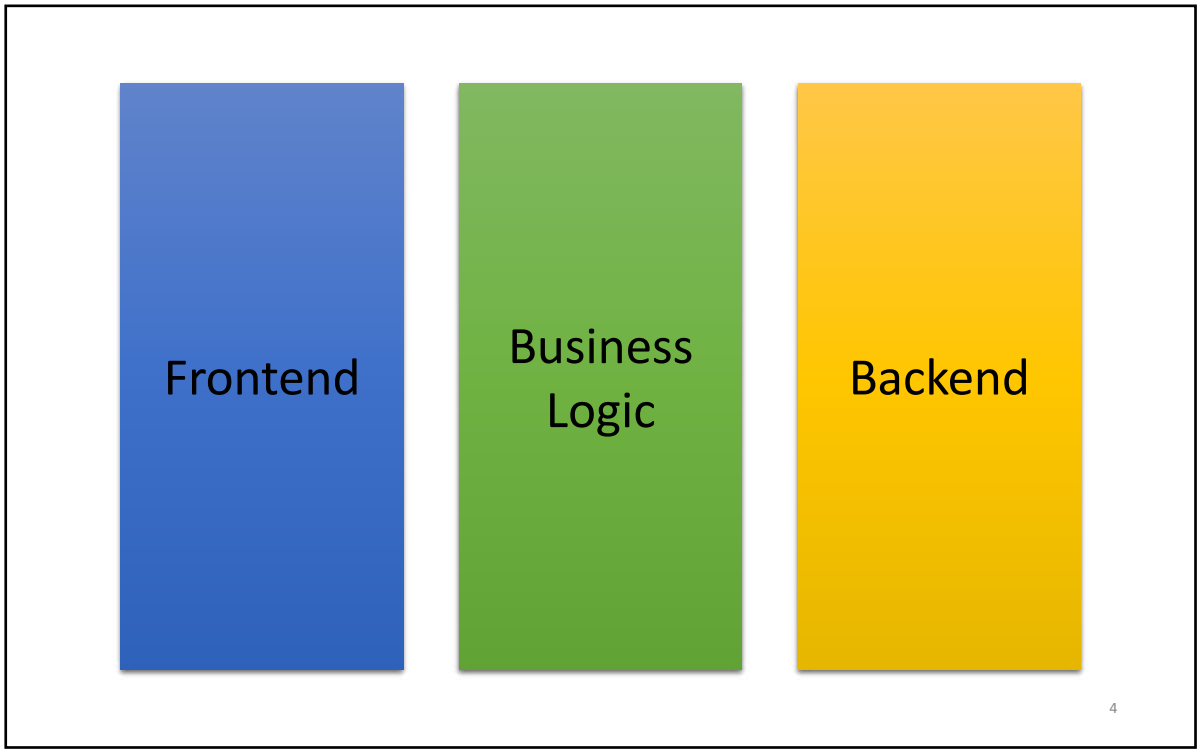
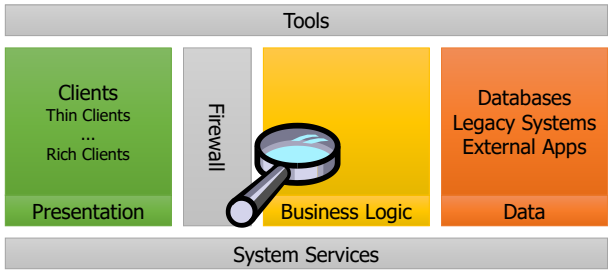

Software
Architectures
for Enterprises
SA4E
Peter Sturm
Universität Trier
Winter 2024


n-Tier Architectures
2


3


Frontend
Business Logic
Backend
4

## Windows DNA



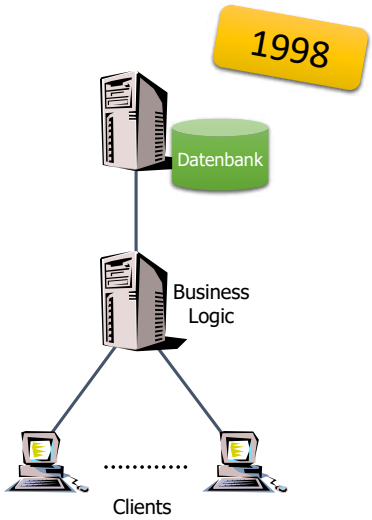

- DNA = Distributed interNet Application Architecture
- 3-Tier Ansatz

45

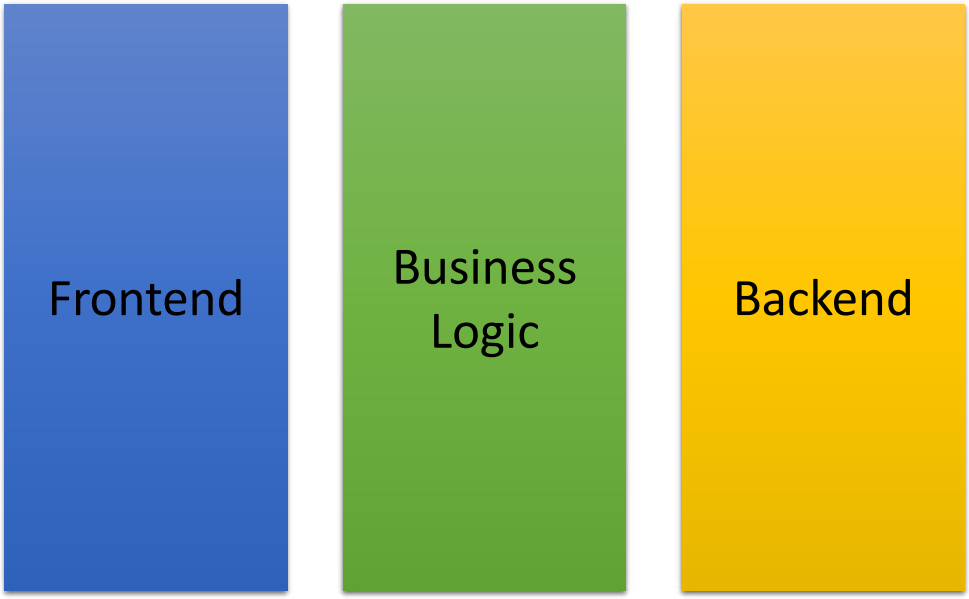
5

## Motivation

- Bean = Komponente
- Zielgruppe
  - Kommerzielle Anwendungen
  - E-Commerce
- Evolution früherer
  - Client/Server-Systeme
  - Transaktionsmonitore
- 3-Tier-Applikationen
  - Client
  - Bussiness Application
  - Database




6




7

## n = 2, 3, 4 oder 5?

- 2
  - Client und Datenbank
  - Transaktionsmonitore
- 4
  - Komplexe UIs (MVC)
  - Integrationsebene (Apache Camel)
- 5 und mehr
  - Caching, Analytics, Authentication
  - Feinere Granularität (Microservices, ...)

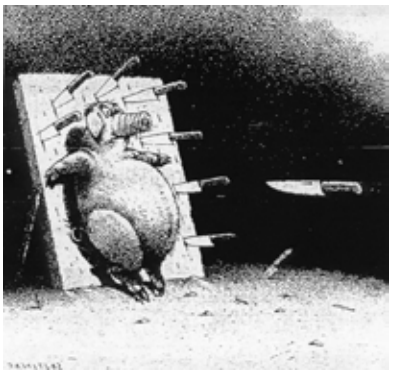
8

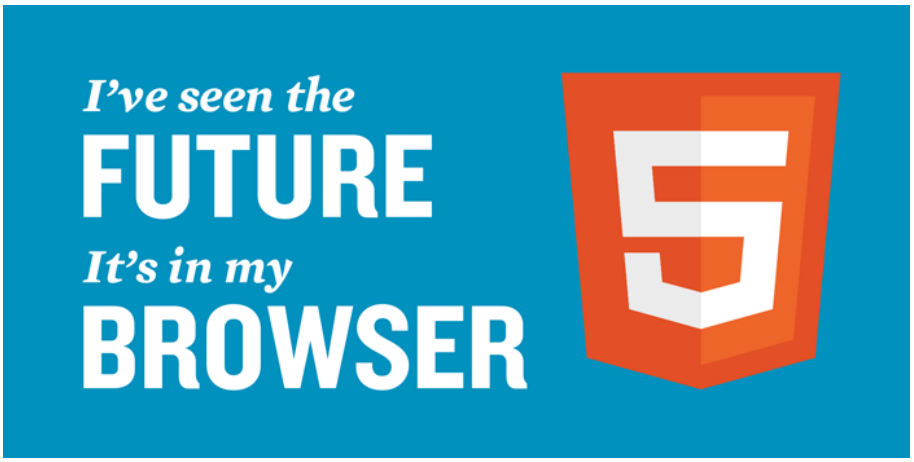
JavaScript rules the world (Midjourney)


JavaScript

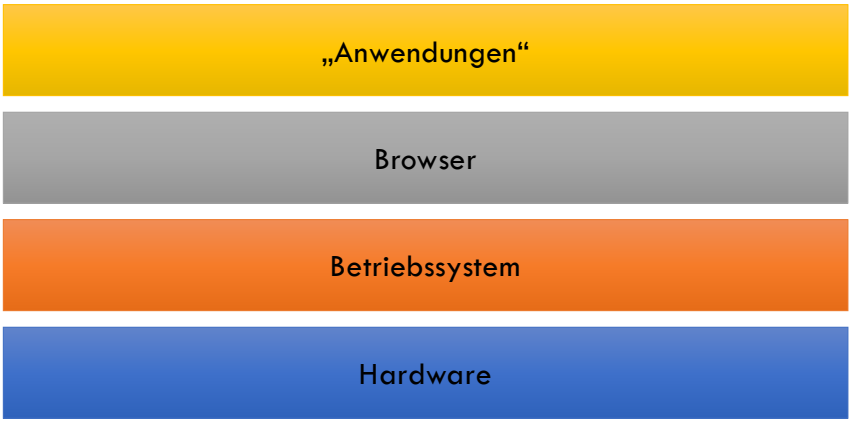
## Disclaimer

- Kein JavaScript-Kurs!
  - Bestimmt können Sie das besser als ich ☺
- Meta-Ebene
- Interaktiv
  - Fragen
  - Ergänzungen
  - Diskussion

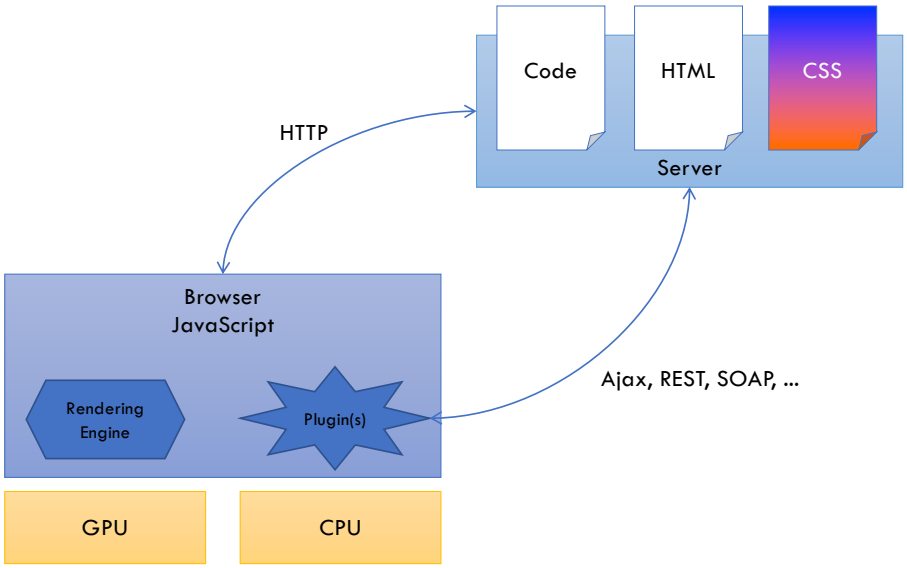


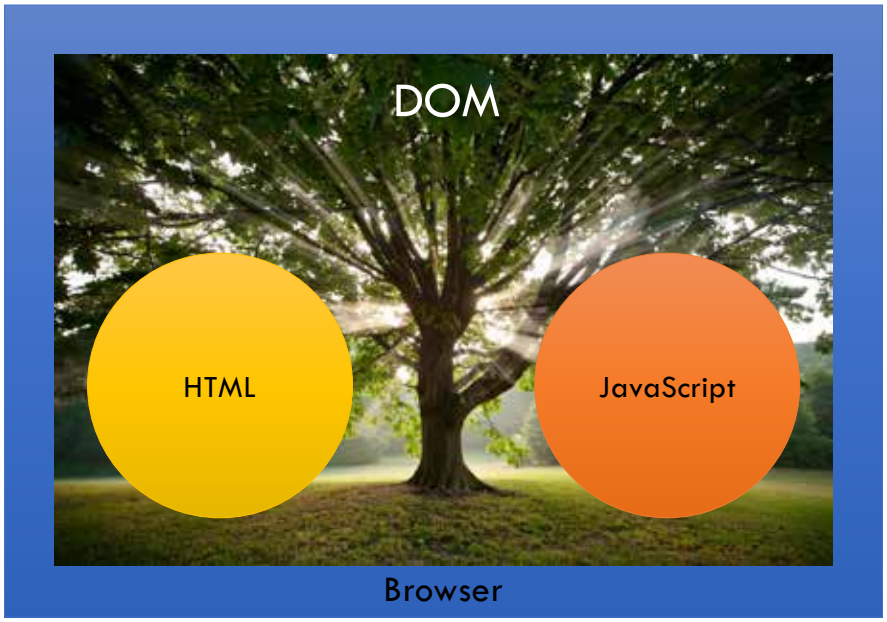


# Browser = OS




# Architektur




DOM
HTML
JavaScript
Browser

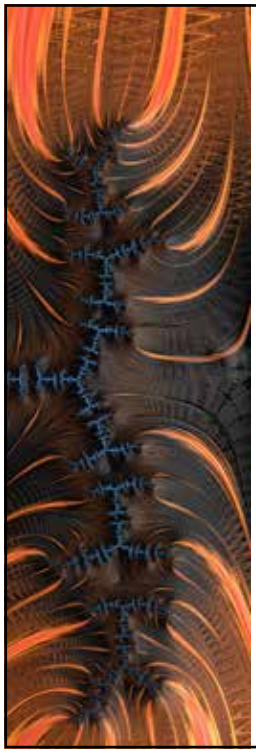
# W3C

- Standards weiten sich immer mehr aus
- Browser-lokaler Storage
- Multi-Threading
  - WebWorker
- Kommunikation über TCP
  - WebSockets

# WebGL

- OpenGL im Browser
- Engines auf WebGL
  - Construct 2
  - pixi.js
  - Three.js
- Browser-Spiele

Epic Citadel (Epic, Mozilla)



- Zugang zu OpenCL
- Heterogeneous Parallel Computing
  - ManyCore
  - GPU
- Vorzugsweise SIMD

# WebRTC

- Real Time Communication
- Audio, Video, beliebige Daten
- JavaScript APIs
  - MediaStream
  - RTCPeerConnection
    - Signal Processing, Codec, Communication, Security, Control bandwidth
  - RTCDataChannel
    - WebSocket API, Geringe Latenz, Unreliable/Reliable, Security

# Leaving the Browser

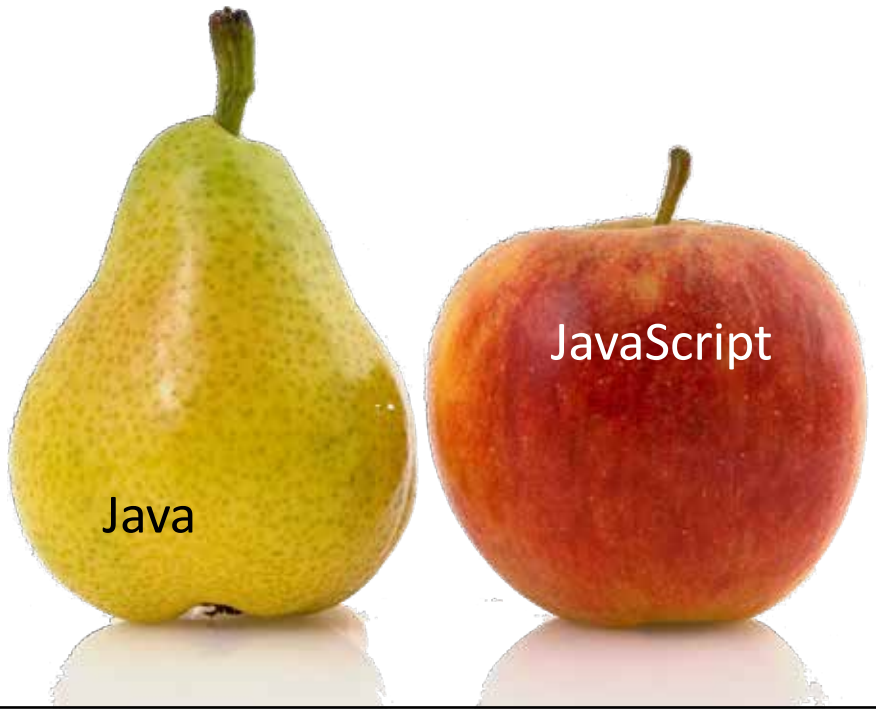
Tessel 2

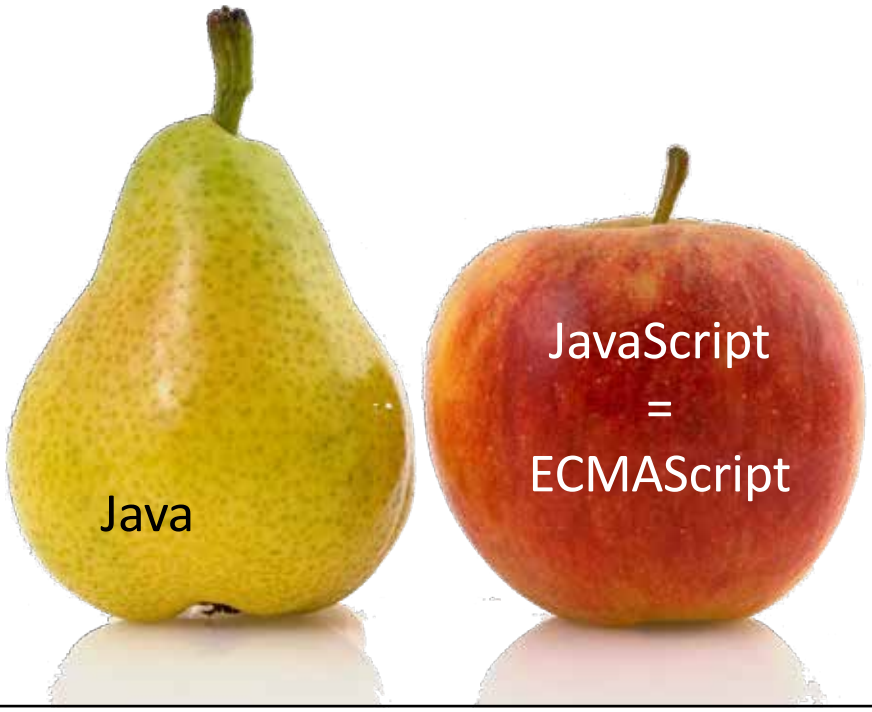
## F(&LF)AQ

- JavaScript?
- The bad parts?
- The good parts?
- JavaScript Patterns?
- Transpiling?
- TypeScript?
- JavaScript @ Backend?
- Frameworks?
- Isomorphes JavaScript?
- Fullstack?
- MEAN?
- JavaScript = Assembler für das Web?

# Historisches

- 1995
- Brendan Eich
- Netscape


Java
JavaScript
=
ECMAScript

Lesetipp


THE PRINCIPLES OF
OBJECT-ORIENTED
JAVASCRIPT
ELOQUENT
JAVASCRIPT
SECOND EDITION
A Modern Introduction
to Programming

# The bad parts

## Risiken

- Keine typstrenge Sprache
- Dynamische Resolution
- Exotisches Objektmodell
  - Klassen-basierte Ansätze sind geläufiger

## Abstraktionsstufe?

- Niedrig, Technisch, teilweise skuril
- Wenig Halt für große Anwendungen
  - Viele Konventionen
  - Vielle Pattern
  - …


Ben Halpern @bhalp1
Sometimes when I'm writing Javascript I want to throw up my hands and say "this is bullshit!" but I can never remember what "this" refers to


The good parts


## Vorzüge

- Keine typstrenge Sprache
- Dynamische Resolution
- Rapid Prototyping
- Implement first, model later (or never)

# JavaScript Pattern

- JavaScript-Umfeld besonders aktiv
  - ECMA-Standards mit Browser-Standards abgleichen
- Neuer Name für Dirty Hacks ☺
- Sprache ist besonders geeignet
- Eigener Fachjargon

# Shims

A **shim** is any piece of code that provides a layer of abstraction or additional functionality to bridge differences between environments. In the context of JavaScript:

- **Purpose:** Shims modify or add APIs that are not natively available in the environment.
- **Scope:** They often act as patches to implement missing functionality.
- **Example:** A shim might overwrite an existing method to fix bugs or ensure consistency across environments.

**Example: Providing Array.prototype.forEach in an environment where it's missing**

javascript Copy

```javascript
if (!Array.prototype.forEach) {
  Array.prototype.forEach = function (callback, thisArg) {
    for (let i = 0; i < this.length; i++) {
      callback.call(thisArg, this[i], i, this);
    }
  };
}
```

40

## Polyfill

A **polyfill** is a specific type of shim that **mimics modern JavaScript features** in older environments that don't support them.

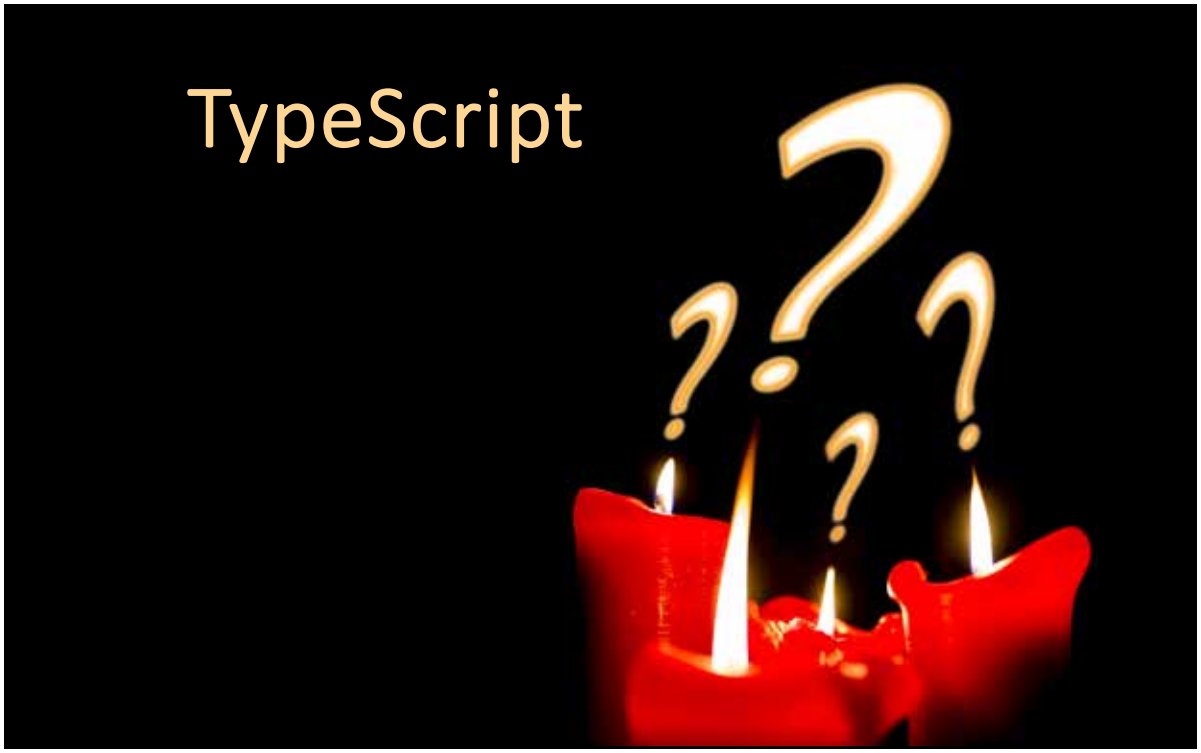
- **Purpose:** Polyfills implement missing APIs or features as closely as possible to the standard.
- **Scope:** They focus on providing modern functionality to environments without it.
- **Example:** A polyfill introduces functionality like `Promise`, `fetch`, or `Object.assign` in environments that lack support.

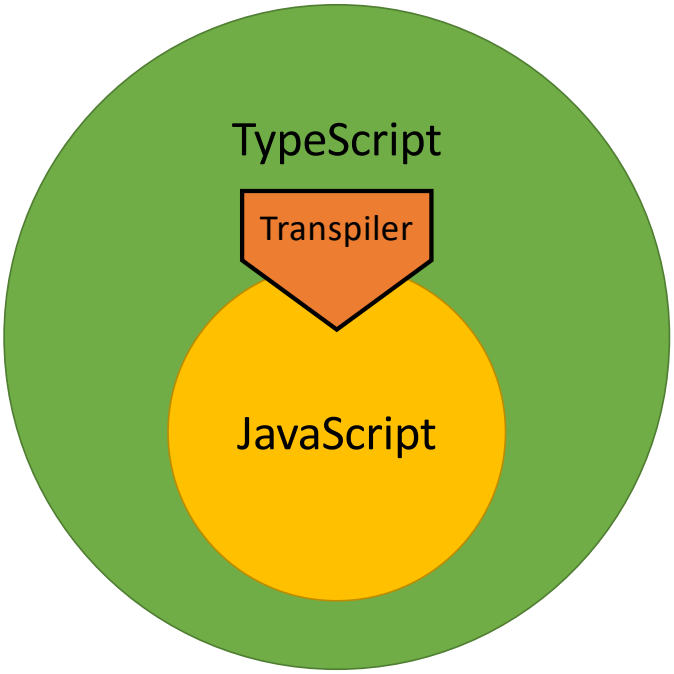
**Example: Polyfill for Object.assign**

javascript Copy

```javascript
if (typeof Object.assign !== 'function') {
  Object.assign = function (target, ...sources) {
    sources.forEach((source) => {
      if (source) {
        Object.keys(source).forEach((key) => {
          target[key] = source[key];
        });
      }
    });
    return target;
  };
}
```

Here, the polyfill ensures `Object.assign` works as expected in older JavaScript engines.






## Best of both worlds

- Verläßlichkeit des Transpilers
  - Fehler früh erkennen
- Flexibilität der Laufzeitplattform

# JavaScript @Backend

**Mario Fusco** @mariofusco 1d

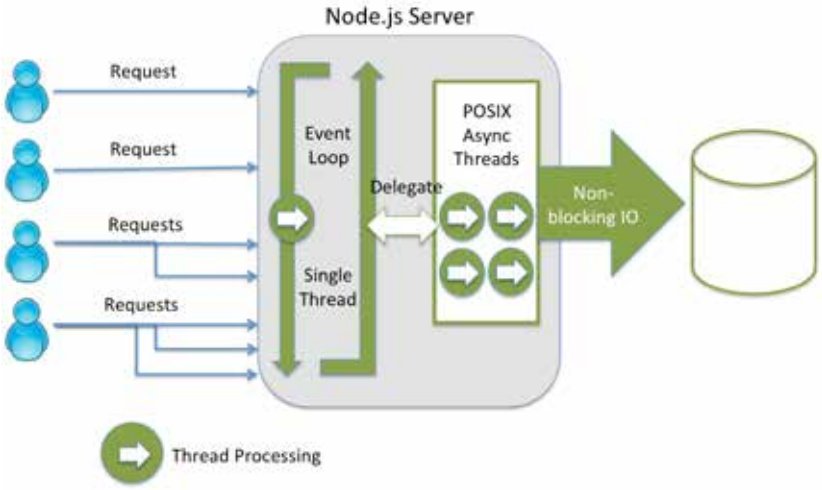
JavaScript on the server side
pic.twitter.com/NSSGdbNbxz

# node.js

- Basiert auf Chrome JavaScript Runtime
- eher Server Framework
- Erweiterbar über Packages
- Package Manager npm

- Alternativen
  - Deno, Bun

# Event Processing



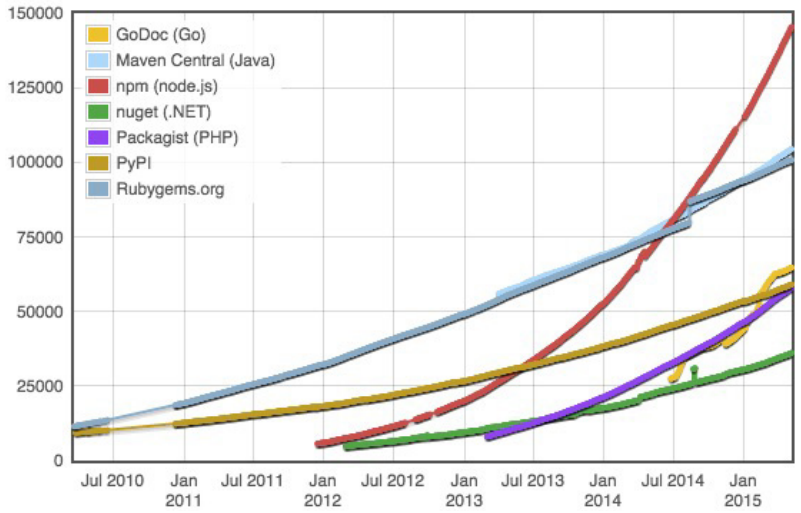
## „Hello World“

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(1337, '127.0.0.1');
console.log('Server running at http://127.0.0.1:1337/');
```

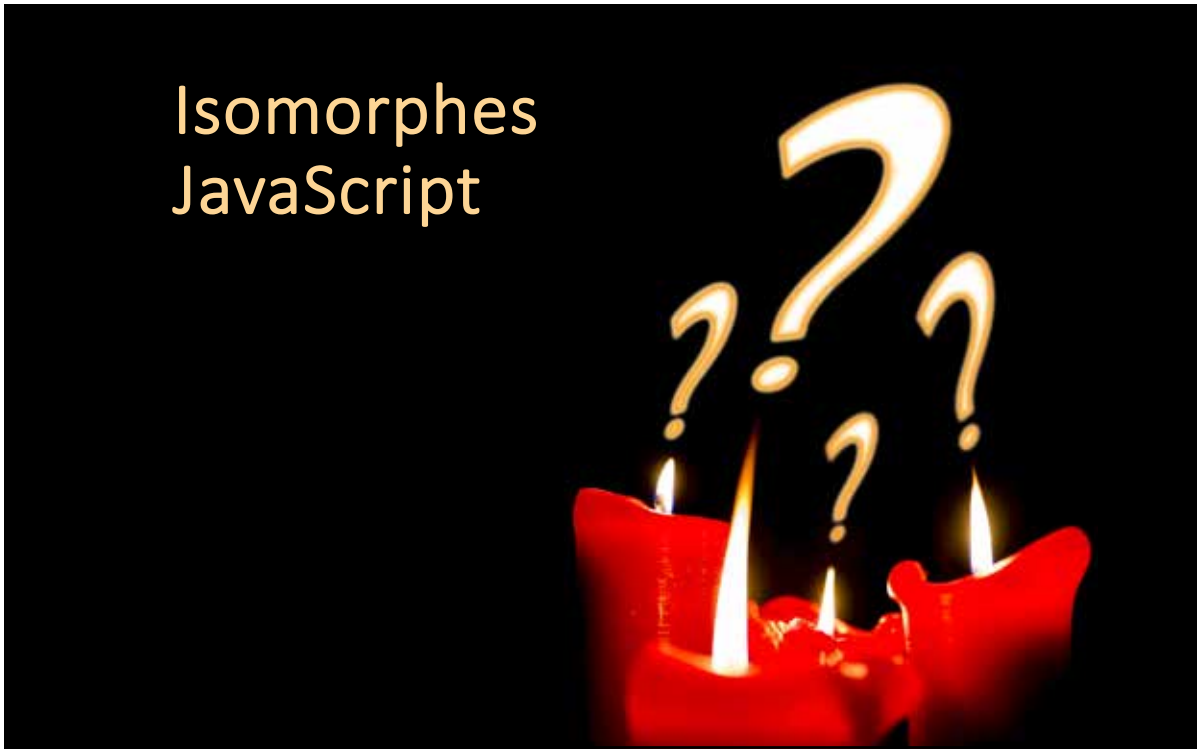
## Packages

- Vielzahl an Zusatzpaketen
  - Mehr als 2 Millionen auf npm
- Kleine Auswahl
  - Cluster (Multicore)
  - Crypto
  - DNS
  - File System
  - HTTP /HTTPS
  - Net
  - Operating System


## Modulzahlen




DAYS WITHOUT
NEW JAVASCRIPT
FRAMEWORK
2
React


Isomorphes
JavaScript

## Isomorph

- “iso” gleich
- “morph” Form/Gestalt

Isomorphism describes that if you look at the same entity in two different contexts, you should get the same thing. Here the contexts are server and client.

Isomorphic.net/javascript

## Vorteile

- LOPE ☺
  - Learn once program everywhere
- Web-Entwickler anderweitig nutzbar
- Synergieeffekte
  - Tools
- Suchmaschinen verstehen serverseitiges JavaScript

## Nachteile

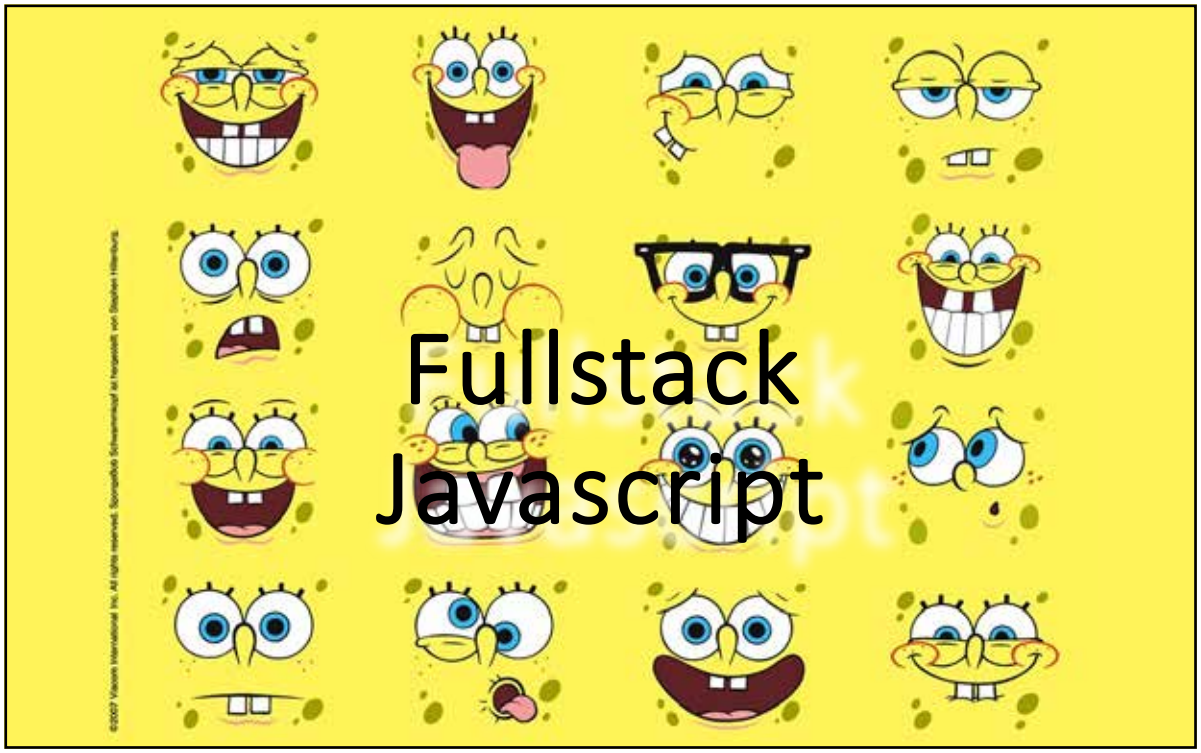
- JavaScript geeignet für Serverseite?
- Overhead
  - Zusätzlicher Interpreter (vgl. PHP)
  - Mehr Speicher- und CPU-Bedarf
  - Performanz?
  - Sicherheit?

## Varianten

- Non opinionated
- Git: Plumbing
- Ruby
  - Sinatra

- Opinionated
- Git: Porcelain
- Ruby
  - ... on Rails


Fullstack


Fullstack
Javascript

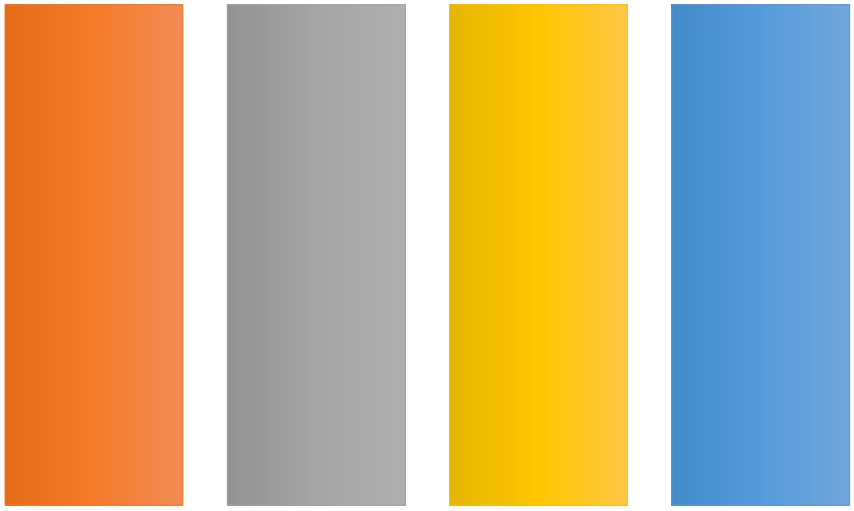
## Fullstack Frameworks

- Next.js
- Nuxt.js
- NestJS
- MEAN/MERN/MEVN
  - Angular, React, Vue
- Meteor
- Sails.js (in Anlehnung an Ruby on Rails)

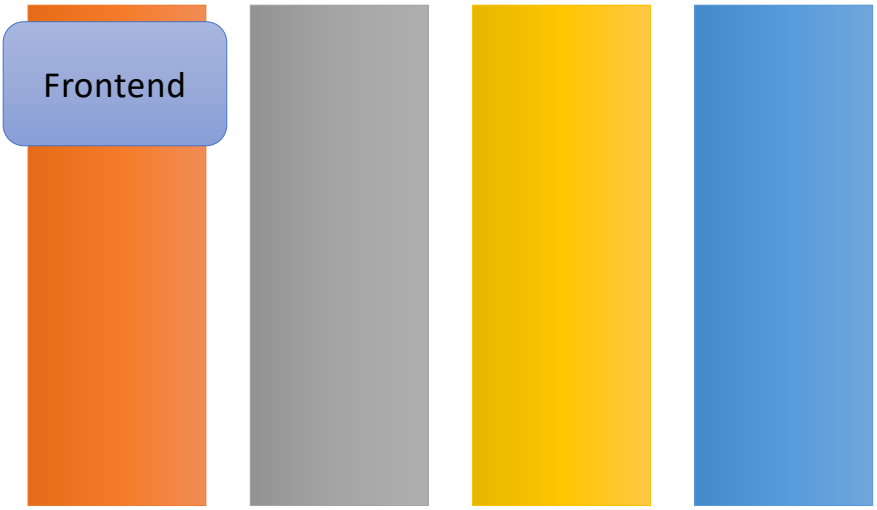
61

## Stack = Vertikal

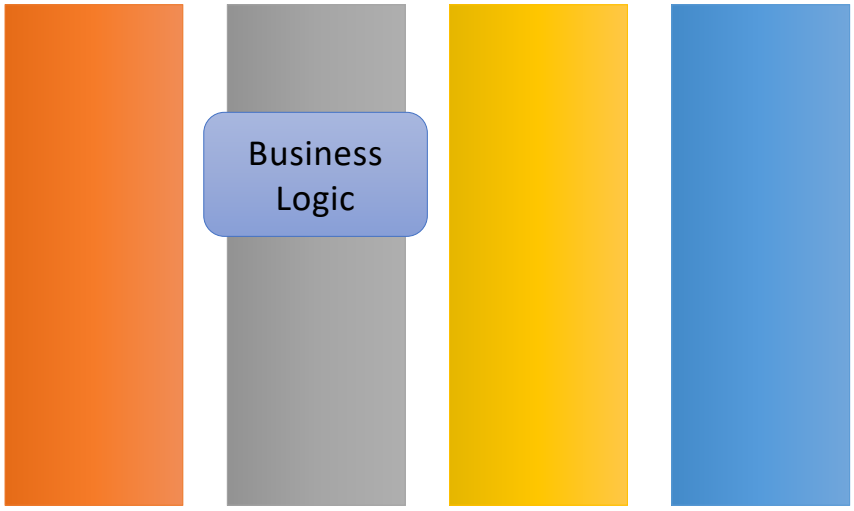
## horizontal = fullstack



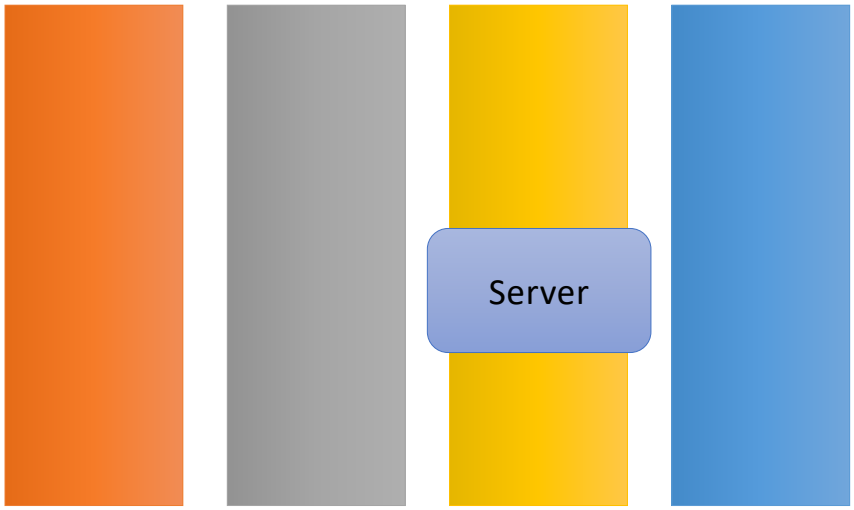
## horizontal = fullstack



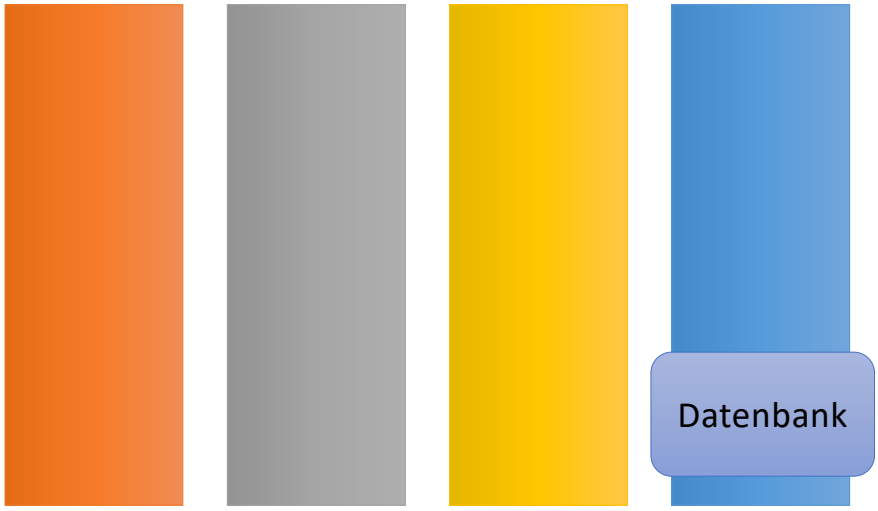
## horizontal = fullstack

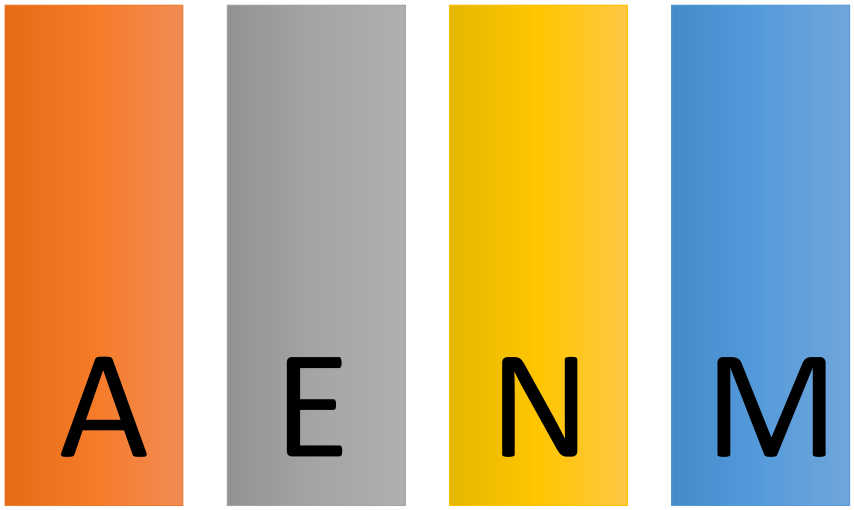



## horizontal = fullstack



horizontal = fullstack

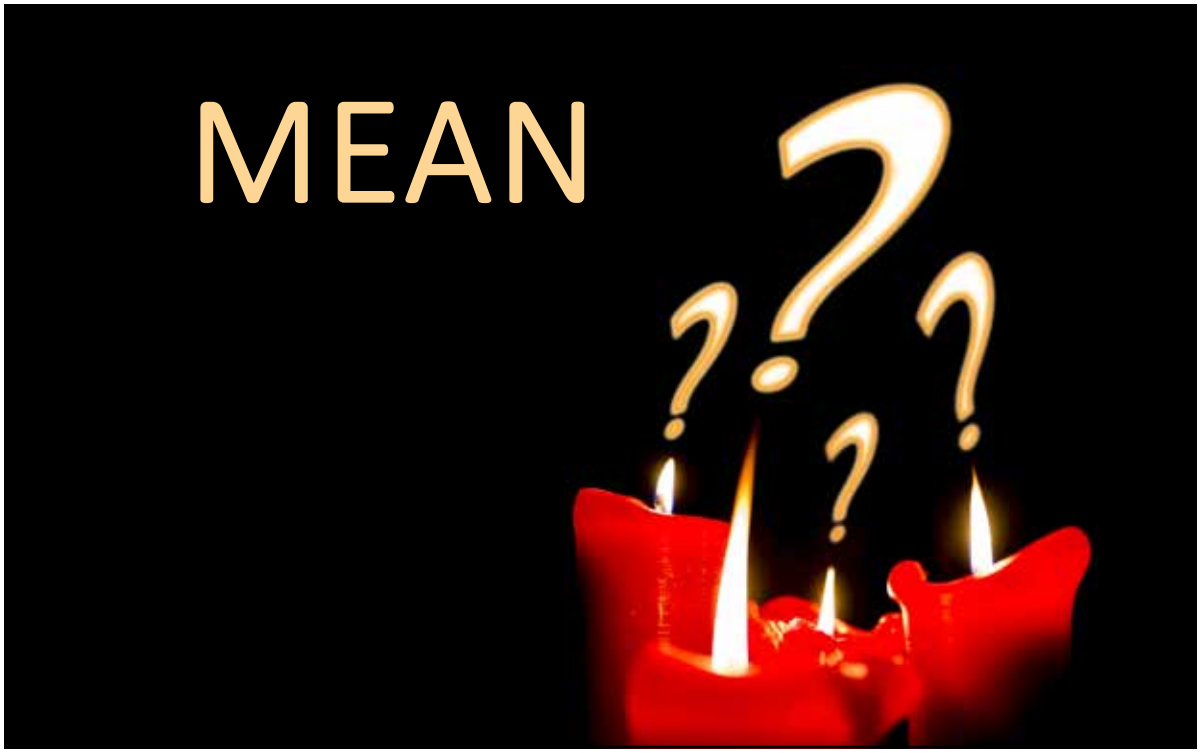


# M E A N

## MEAN

- Angular.js
  - Frontend / Web Client
- Express.js
  - Serverseitiges Framework
  - Template Engine
- Node.js
  - WWW, Ajax, REST
- MongoDB
  - Document Store




## mongodb

- Horizontal skalierbare NoSQL-Datenbank
- Document Store
- Dokumentformat JSON / BSON
- Unterstützt sekundäre Indizies
- JavaScript ist Query Language
- Mongoose
  - Objektmodellierung für Node.js

## express.js

- Server Setup
- „Routing URLs to Responses“
- HTML templating engines
- User and session support

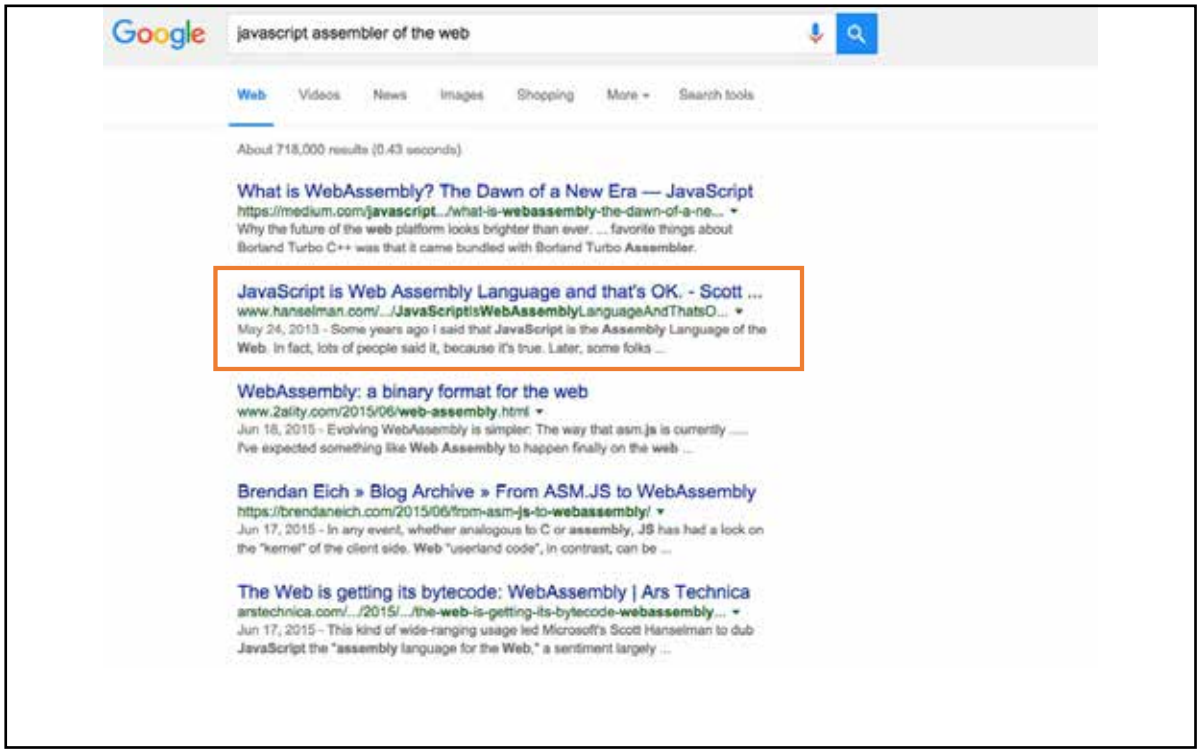
```
// Import Express.js
const express = require('express');
const app = express();
const port = 3000;

// Middleware to parse JSON request bodies
app.use(express.json());

// In-memory "database" for to-do items
let todos = [
  { id: 1, task: 'Learn Express.js', completed: fa
  { id: 2, task: 'Build a REST API', completed: fa
];

// GET: Fetch all to-dos
app.get('/todos', (req, res) => {
  res.json(todos);
});

// POST: Add a new to-do
app.post('/todos', (req, res) => {
  const newTodo = {
    id: todos.length + 1,
    task: req.body.task,
    completed: false,
  };
  todos.push(newTodo);
  res.status(201).json(newTodo);
});

// PUT: Update a to-do (mark as completed)
app.put('/todos/:id', (req, res) => {
  const todo = todos.find(t => t.id === parseInt(r
  if (!todo) return res.status(404).json({ error:

  todo.completed = req.body.completed ?? todo.comp
  res.json(todo);
});
```

## angular.js

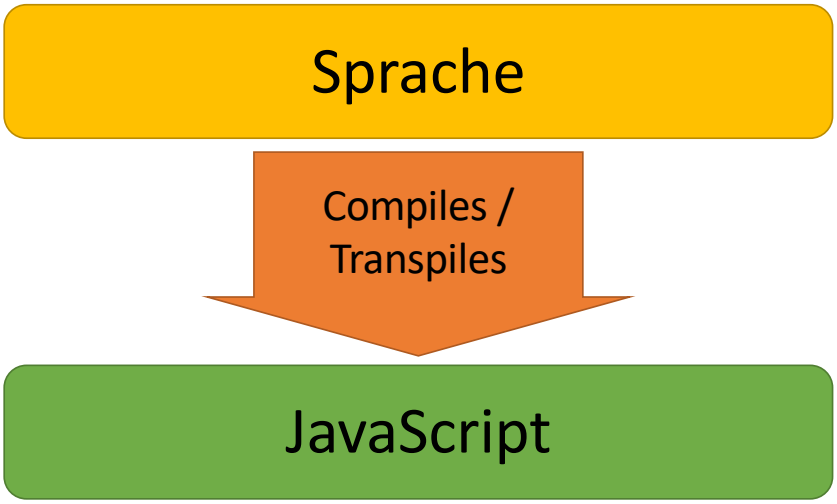
- MVC Framework
  - Flexibel bei Frontend/Backend-Zuordnung
- Bindings über Tags in HTML
- Single Page Applications (SPA)

```
<body ng-controller="TodoController">

  <h1>To-Do List</h1>
  <input type="text" ng-model="newTask" placeholder=
  <button ng-click="addTask()">Add</button>

  <ul>
    <li ng-repeat="task in tasks">
      {{ task.name }}
      <button ng-click="removeTask($index)">Remove</
    </li>
  </ul>

</body>
<script>
  // Define the AngularJS app
  var app = angular.module('todoApp', []);

  // Define the controller
  app.controller('TodoController', function ($scope
    // Initialize the task list
    $scope.tasks = [
      { name: 'Learn AngularJS' },
      { name: 'Build a to-do app' }
    ];

    // Add a new task
    $scope.addTask = function () {
      if ($scope.newTask) {
        $scope.tasks.push({ name: $scope.newTask });
        $scope.newTask = ''; // Clear input field
      }
    };
```

JavaScript
=
Assembler für
das Web



## Gibt es das?




## Ja, gibt es!

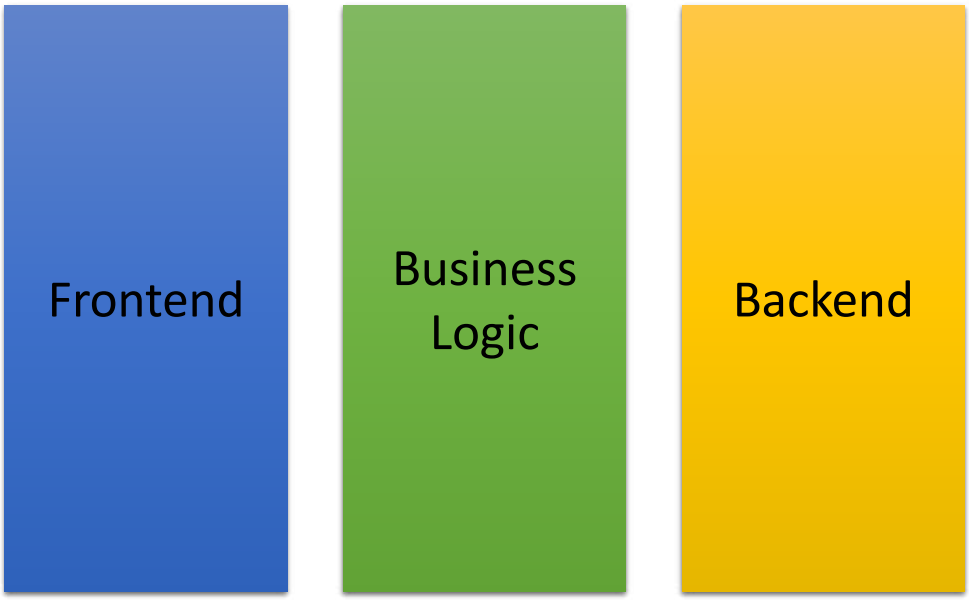
- Web Ubiquity: Alle Browser unterstützen JavaScript
- Performante JavaScript engines (Chrome V8, Firefox SpiderMonkey)
- Spracherweiterungen
  - Type safety
  - Immutability
  - Functional programming

78

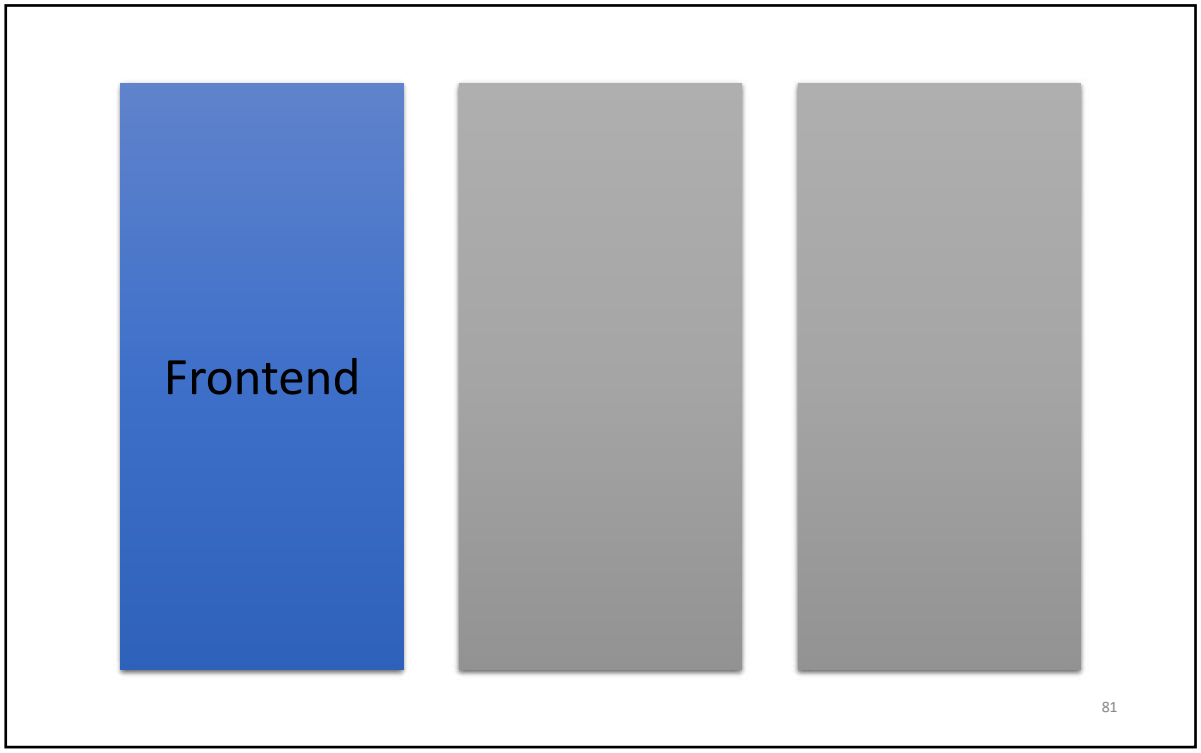
## Beispiele

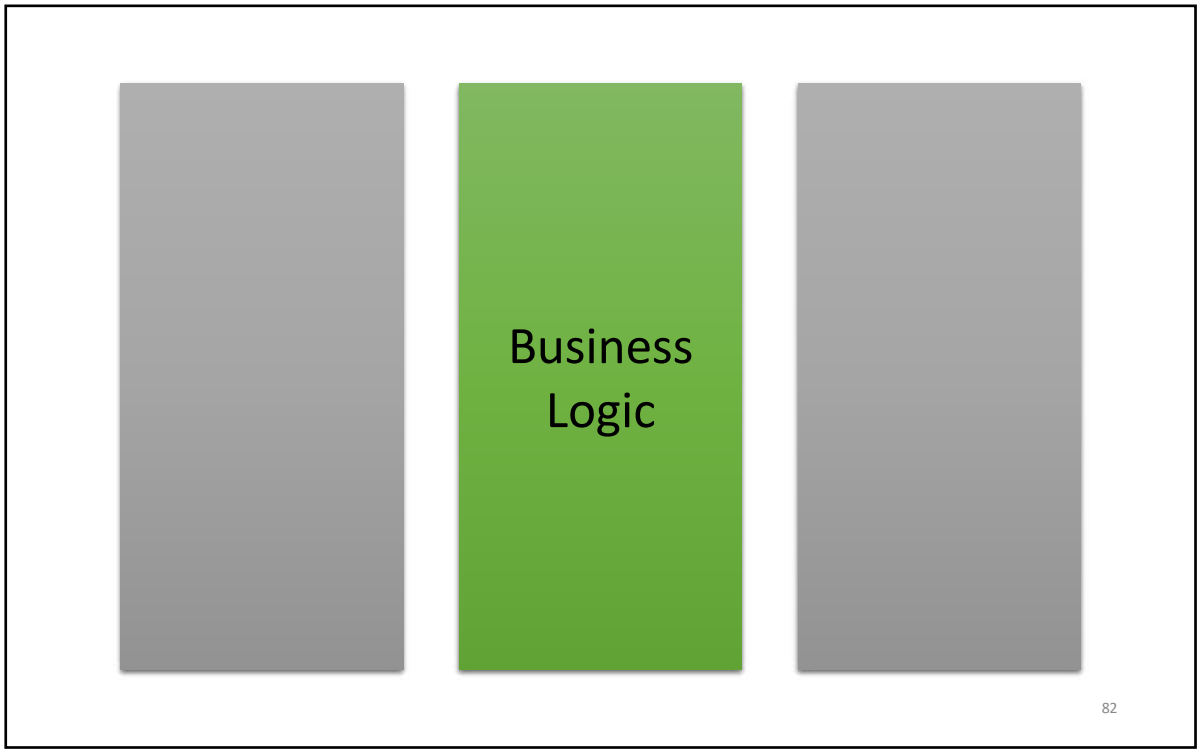
- TypeScript
- Dart (Google)
- CoffeeScript
- ClojureScript
- Elm
- Kotlin/JS
- Scala.js
- Rust

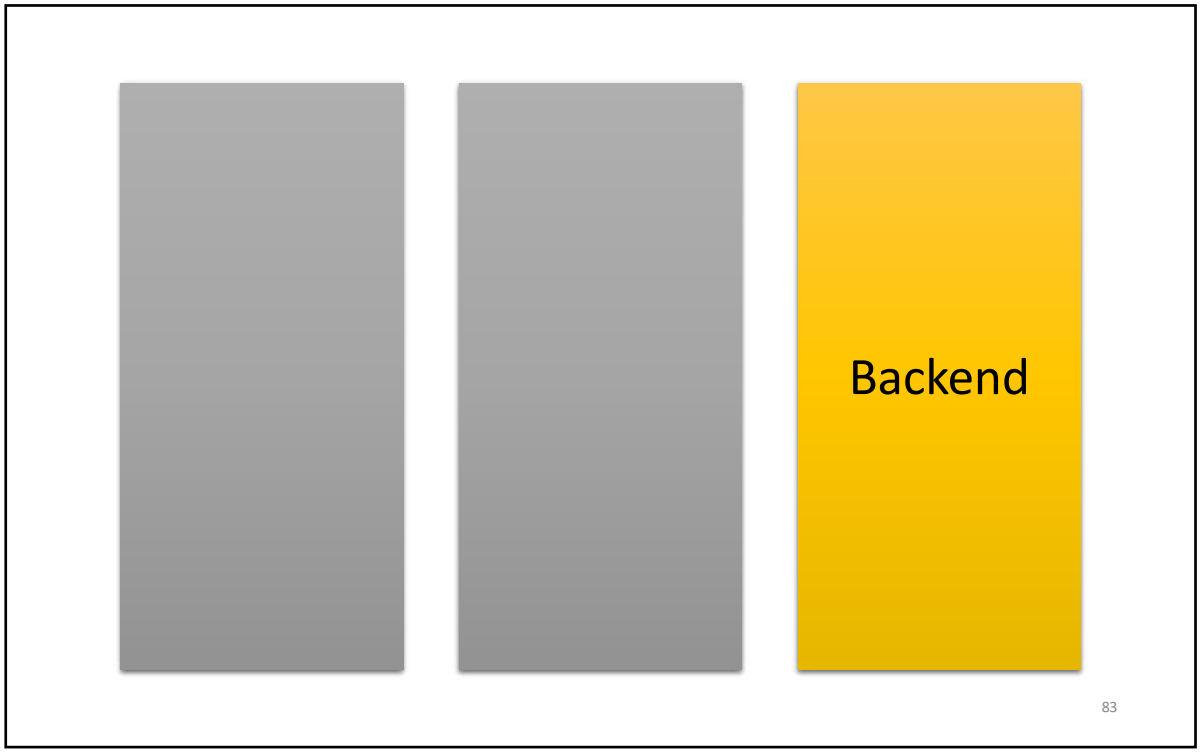
79




80


Frontend
81


Business
Logic
82


Backend
83