Sicheres Generieren auf dem Webserver

- private Schlüssel sollten niemals unsicher übertragen werden
- Bei der Anfrage kann eine sog. „Certificate Signing Request“ (CSR) generiert
  - Enthält folgende Informationen
    - Land
    - Bundesland
    - Stadt
    - Organisation
    - Abteilung
    - Common Name (Bezeichnung / URL)
    - Mail-Adresse
  - asymmetrischer privater RSA-Schlüssel verbleibt auf dem Server

CSR generieren

- Ein CSR kann bspw. mit OpenSSL generiert werden
- Die Anfrage wird anschließend bei der CA für die Generierung eines Zertifikats verwendet

```
openssl req -nodes
         -newkey rsa:4096
         -sha256
         -keyout 'meinedomain_com.key'
         -out 'meinedomain_com.csr'
         -subj '/CN=meinedomain.com/C=DE'
```

/dpetrasch/DHBW

Installation auf dem Webserver (nginx)

- Dem Webserver muss mitgeteilt werden, dass eine Website via SSL mit einem bestimmten Zertifikat bereitgestellt werden soll
- Port ist TCP 443
- Protokolle sind unterschiedlich
- Anpassen der Config unter „/etc/nginx/sites-available/meinedomain.com“

```
# Domain-Name
server_name meinedomain.com www.meinedomain.com;

## SSL / TLS
listen 443 ssl;

## Protokolle - hierzu spaeter mehr
ssl_protocols TLSv1 TLSv1.1 TLSv1.2;

## Öffentlicher und privater Schluessel
ssl_certificate        /etc/ssl/meinedomain.com.public;
ssl_certificate_key    /etc/ssl/meinedomain.com.private;
```

/dpetrasch/DHBW

Installation auf dem Webserver (Apache)

- Dem Webserver muss mitgeteilt werden, dass eine Website via SSL mit einem bestimmten Zertifikat bereitgestellt werden soll
- Port ist TCP 443
- Protokolle sind unterschiedlich
- Anpassen der Config unter „/etc/apache2/sites-available/ssl.conf“

```
<VirtualHost *:443>
   SSLEngine on
   SSLCertificateFile     /etc/ssl/meinedomain.com.public
   SSLCertificateKeyFile  /etc/ssl/meinedomain.com.private
   # Pfad zu den Webinhalten
   DocumentRoot           /var/www/meinedomain.com/htdocs/
</VirtualHost>
```

/dpetrasch/DHBW

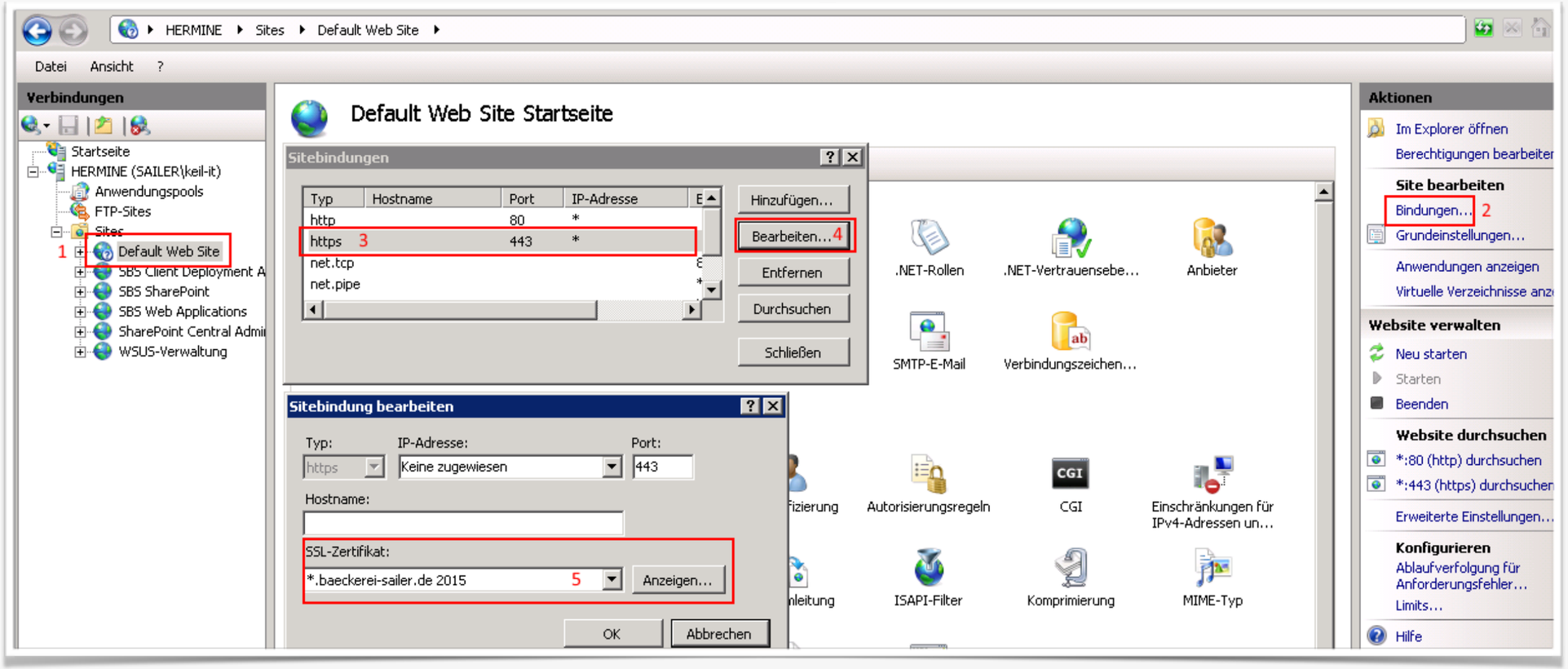
Installation auf dem Webserver (Apache)

- Dem Webserver muss mitgeteilt werden, dass eine Website via SSL mit einem bestimmten Zertifikat bereitgestellt werden soll
- Port ist TCP 443
- Protokolle sind unterschiedlich
- Anpassen der Config unter „/etc/apache2/sites-available/ssl.conf“

```
<VirtualHost *:443>
   SSLEngine on
   SSLCertificateFile     /etc/ssl/meinedomain.com.public
   SSLCertificateKeyFile  /etc/ssl/meinedomain.com.private
   # Pfad zu den Webinhalten
   DocumentRoot           /var/www/meinedomain.com/htdocs/
</VirtualHost>
```

/dpetrasch/DHBW

Installation auf dem Webserver (Windows IIS)



trim/modbes

Welche Protokolle sollen verwendet werden?

- SSL ist mehr als nur ein Protokoll, sondern vielmehr eine Familie von Protokollen
- Unterstützung von Fallback für altere Systeme
- Verschiedene Plattform erfordern unterschiedliche Protokollstandards

Poodle

- Wo?

  Ist ein Fehler im (veralteten) SSLv3-Protokoll

  Sicherheitslücke ist seit 1999 anwendbar.

- Warum schlimm?

  Verschlüsselte Datenpakete können durch eingeschleusten Javascript-Code entschlüsselt werden. Dies kann auch durch Dritte als „Man-in-the-middle“-Attacke durchgeführt werden.

- Wie schützen?

  Serverseitig SSLv3 Protokoll deaktivieren, bzw. TLS_FALLBACK_SCSV verwenden.

Heartbleed

- Wo?

  Ist ein Fehler in OpenSSL 1.0.1 bis 1.0.1f

  und wurde ab 1.0.1g behoben

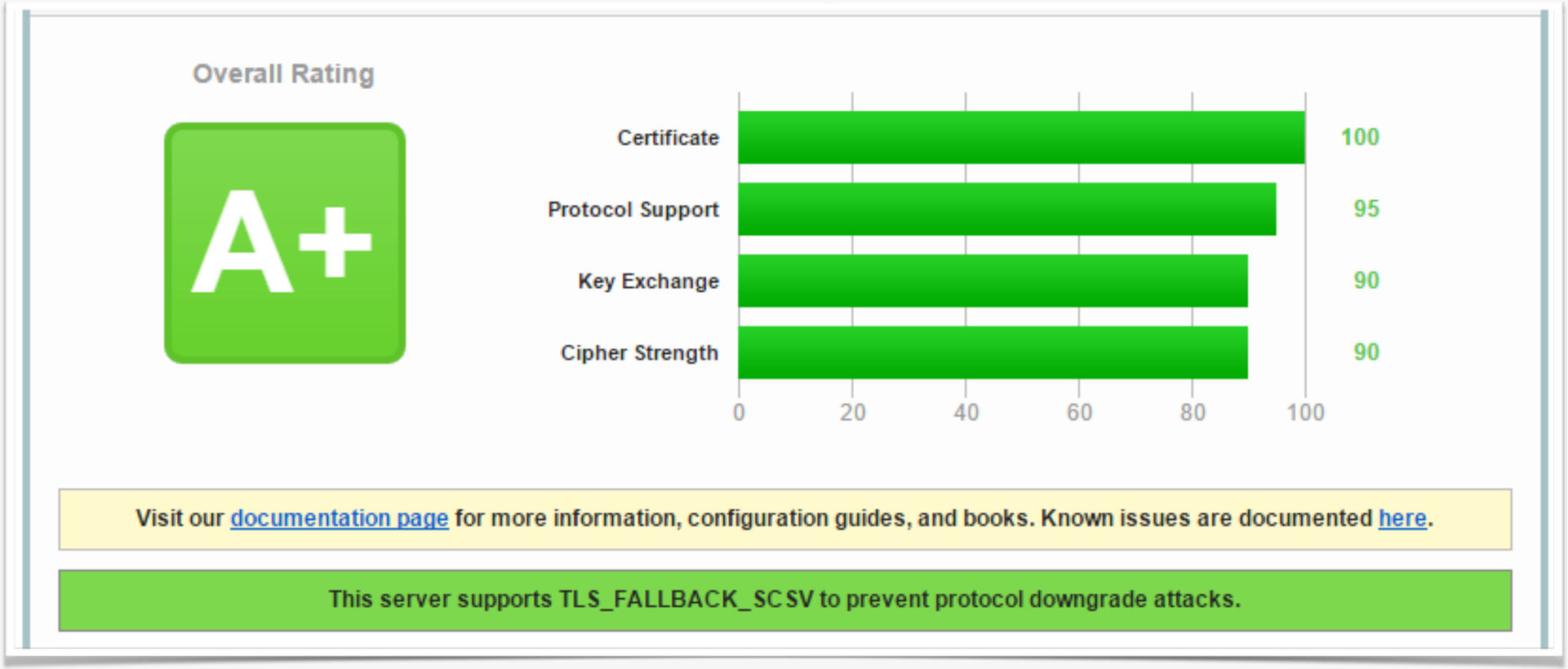
- Warum schlimm?

  Ursprüngliche „Heartbeat“ (periodische Kontaktaufnahme) Funktion kann überlastet werden und es können Teile des aktiven Arbeitsspeichers (bis zu 64kb) ausgelesen werden. Wiederholt können somit Benutzerpasswörter und private Schlüssel abgerufen werden.

- Wie schützen?

  OpenSSL serverseitig aktualisieren.

SSL Verbindungstest (Qualys SSL Labs)



tr.im/ssltest

Good to know: StartSSL und Let's Encrypt

- Kostenlose Web- und SMIME-Zertifikate stellt StartSSL aus
  - Ebenfalls möglich: günstige Code-Signing-Certificates und Wildcard-SSL
- Let's Encrypt ist eine neue - kostenlose - CA.
  - automatische Verlängerung mit einem Serverdienst
  - kostenfrei
  - automatische Installation und CSR

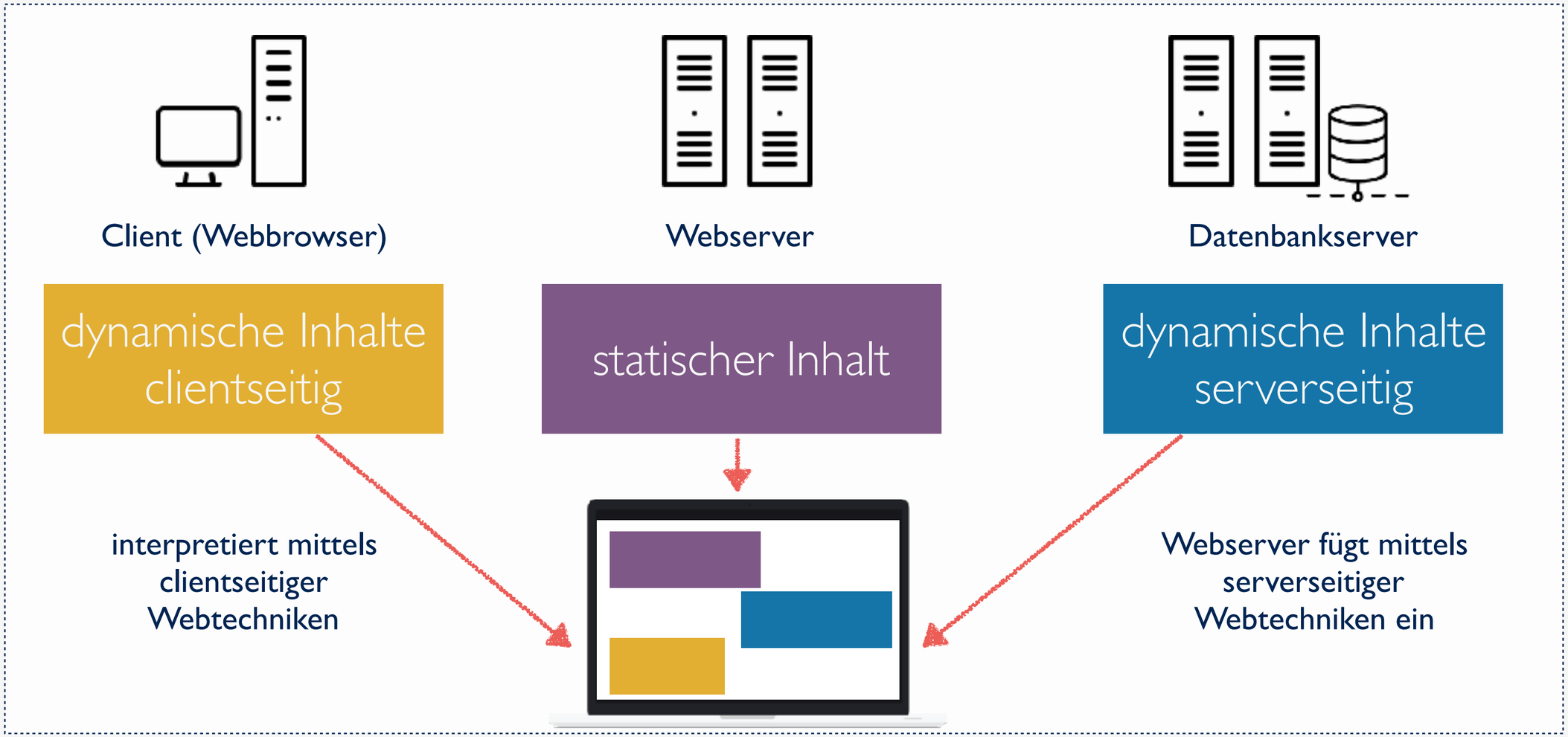
Serverseitige Programmierung

Anfragen dynamisch beantworten

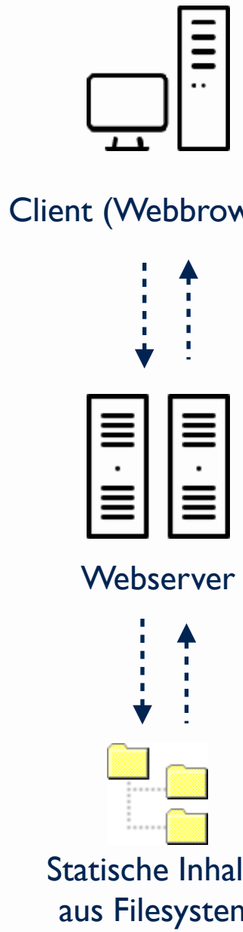
Serverseitige Programmierung

- bedeutet, dass serverseitige Programme erstellt und ausgeführt werden, um
  - Daten aus Datenbanken zu lesen,
  - Daten in Datenbanken zu schreiben und
  - Webseiten dynamischen aufzubauen,
- bevor die Webseite zum Benutzer geschickt wird

Serverseitige Programmierung



Auslieferung statischer Web-Inhalte

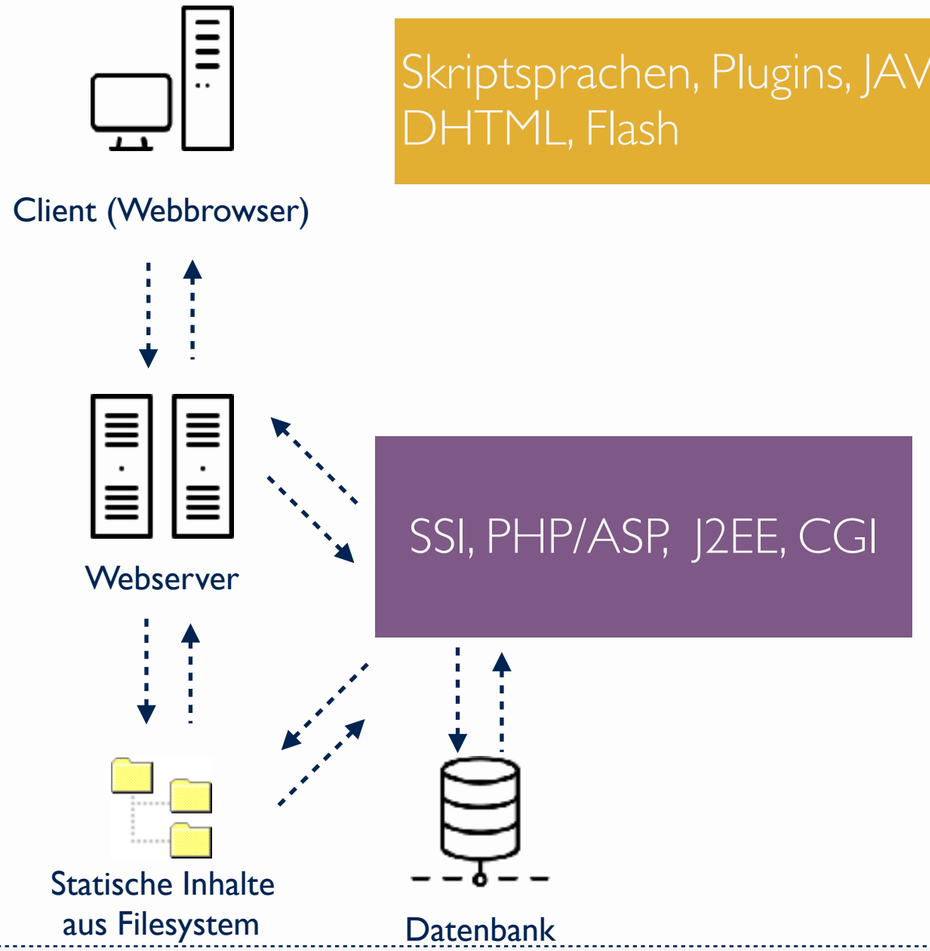


Anfrage an den Webserver
durch Hyperlink oder Direktaufruf

Request / Response im HTTP-Protokoll
- Aufbau TCP-Verbindung
- HTTP-Request, der vom Web-Deamon entgegengenommen wird
- Zurücksenden der Informationen über dieselbe Verbindung
- Schließen der Verbindung

Darstellung auf Client
durch Interpretation von HTML / CSS

Auslieferung dynamischer Web-Inhalte



client- und serverseitige Erweiterungen machen dynamische Webinhalte möglich

je nach Anwendungszweck soll überlegt werden, welche dynamischen Inhalte clientseitig und welche serverseitig realisiert werden

Dynamische Webtechniken

- Erweitern statische Webseiten um dynamische Inhalte
- Clientseitig
  - Bewegung im Browser (Animation)
  - Interaktive Elemente
- Serverseitig
  - Generieren der Seite erst bei Aufruf
  - Integration aktueller Daten aus Datenbanken und anderer Systemen
  - Einbinden speziell für diesen Abruf relevanter Daten

Serverseitige Programmierung

- Zur serverseitigen Programmierung stehen mehrere konkurrierende Technologien zur Verfügung
- Unterschiede in der Leistungsfähigkeit und unterstützten Plattformen
- Wichtige Vertreter
  - CGI und Perl
  - PHP
  - ASP und ASP.NET
  - Coldfusion
  - Java-Servlets und JSP
  - Node.JS

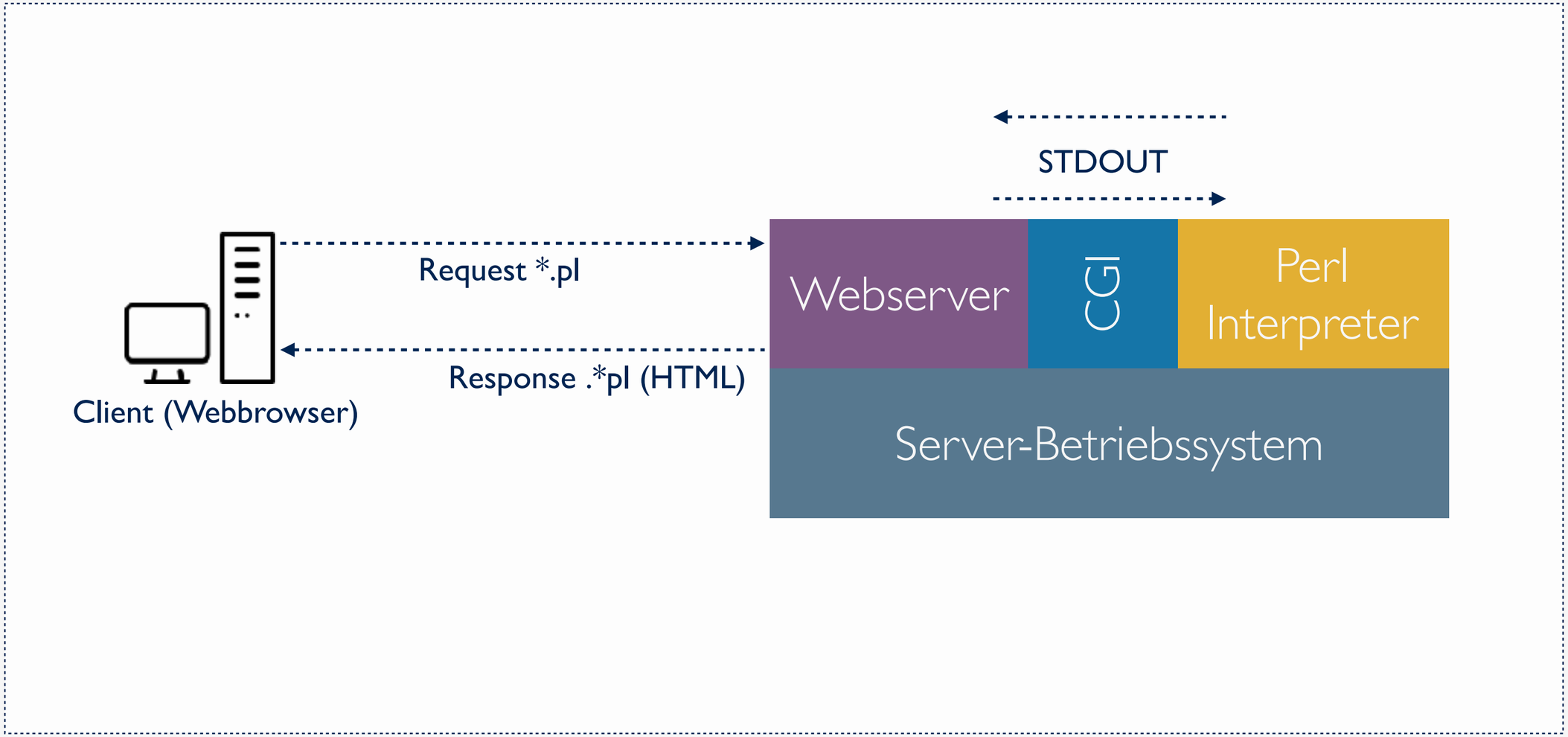
Serverseitige Programmierung: CGI

- CGI = Common Gateway Interface
- CGI definiert eine Schnittstelle zwischen dritter Software und Webserver, die es ermöglicht, mit Programmen auf dem Webserver Anfragen vom Browser zu bearbeiten und Webseiten dynamisch zu generieren
- CGI kann dazu die Eingaben von HTML-Formularen verarbeitenCGI startet für jede Anfrage einen eigenen Prozess auf dem Server, weshalb die Performance des Webservers bei vielen Anfragen sinkt

Serverseitige Programmierung: CGI

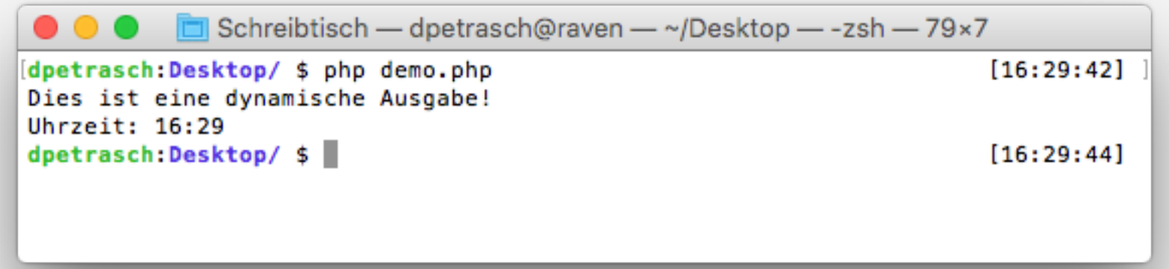
- Perl ist eine Skriptsprache, die zur Laufzeit interpretiert wird und daher als Interpretersprache keinen Compiler benötigt.
- Perl setzt nur einen auf dem Server installierten Perl-Interpreter voraus
- Perl hat Ähnlichkeiten zur C-Syntax und wurde ursprünglich für Unix-Plattformen entwickelt, um schnell kleine Programme für die Netzwerkentwicklung schreiben zu können
- Perl besteht aus einfachen ASCII-Zeichen
- Perl ist frei erhältlich (GPL) es gibt eine kleinere, aber recht rege Perl-Fan-Gemeinde
- Perl wird oft im Zusammenhang mit CGI verwendet, obwohl Perl ursprünglich nicht speziell für CGI entwickelt wurde

Funktionsweise von Perl in Verbindung mit CGI



Ausführung CGI = Aufruf über CLI

- Ausführung auch über die Konsole möglich



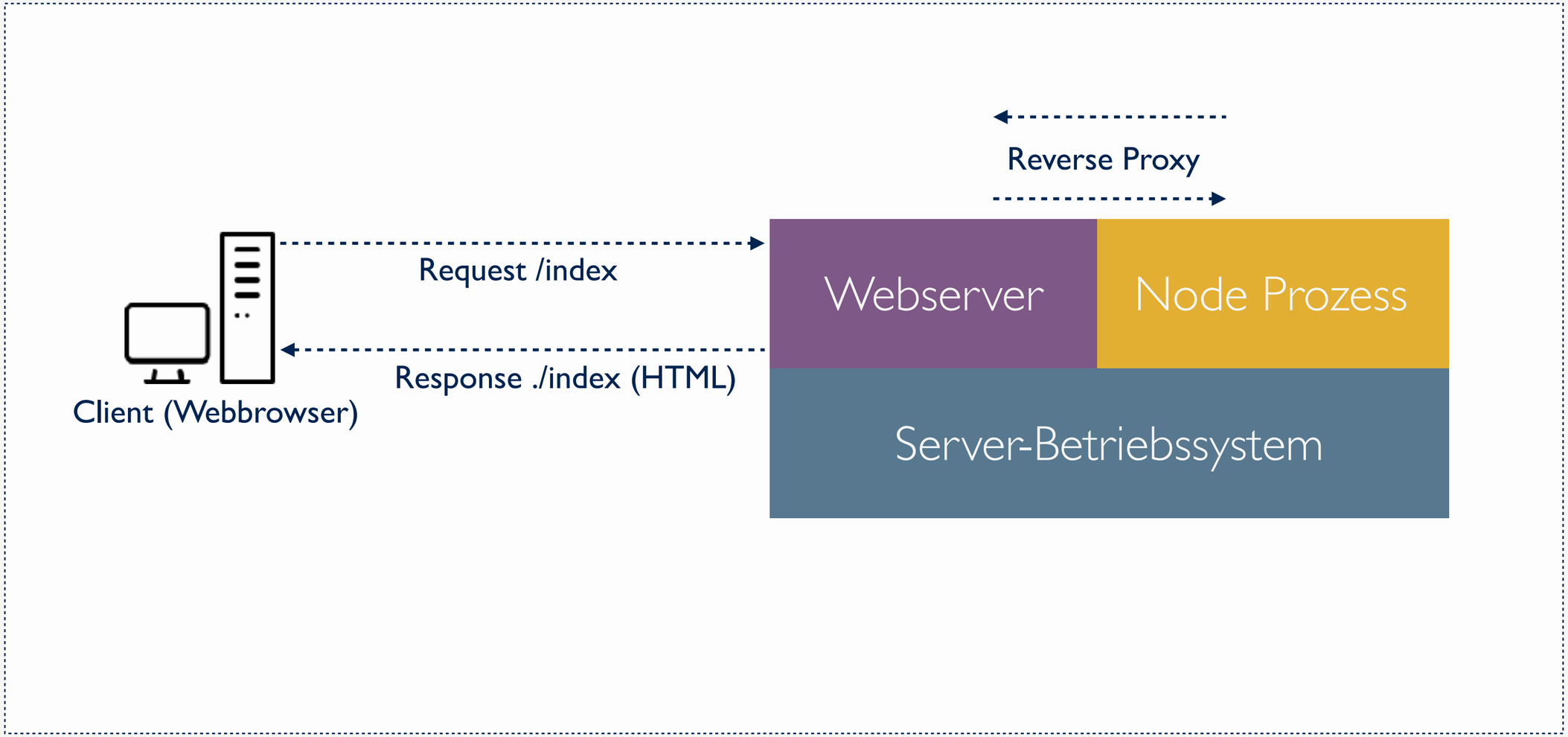

```php
<?php
    echo "Dies ist eine dynamische Ausgabe! \n";

    date_default_timezone_set("Europe/Berlin");

    $timestamp = time();
    $uhrzeit = date("H:i",$timestamp);

    echo "Uhrzeit: " . $uhrzeit . "\n";
?>
```

/dpetrasch/DHBW

Serverseitige Programmierung: Node.js

- Node.js ist eine Laufzeitumgebung zur Ausführung verschiedenen Diensten
- Ausführung von serverseitigem Quellcode erfolgt in einem Prozess, der auch weitere Prozesse starten kann
- Es existieren Module für HTTP-Webserver und WebSocket-Server, die als Basis für die Entwicklung eines Webprojekts verwendet werden
- Anfragen werden von einem Webserver (bspw. Apache oder nginx) lediglich an den Node-Prozess weitergegeben
- Der Webserver fungiert hier als Reverse-proxy

Funktionsweise von Node.js und einem Reverse-Proxy



Cloud

dynamische Lastenverteilung und Ressourcenaufteilung

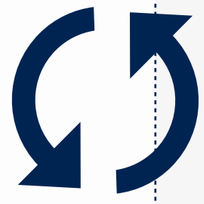
Beispiel: Dokumentenverarbeitung als SaaS

- Texterkennung innerhalb von PDF-Dokumenten als „Software-as-a-Service“
  - Eine eigene Schnittstelle (API) stellt Funktionen für den Upload und Verarbeitung eines PDF-Dokuments bereit
  - Dokument wird zwischengespeichert
  - OCR wird auf das Dokument angewendet
  - PDF-Dokument wird optimiert
  - Abrechnung
  - Bereitstellung des Ergebnisses

SaaS: Mit eigener Infrastruktur

Anfrage vom Client

1. Annahme der Datei
2. Zwischenspeichern der Datei und Einsortieren in eine Verarbeitungsschlange
3. Starten der Erkennung des Dokuments
4. ….Warten ….
5. …. Warten ….
6. Ergebnis abspeichern und an den Client zurücksenden



Antwort an den Client

7. nächste Datei verarbeiten

Problem tritt auf, Dienst beendet sich
Verarbeitung nicht mehr möglich

Clouddienste

- Anfragen im Web werden nicht von einem dedizierten Server sondern von einem Gesamtsystem verarbeitet
- Das Gesamtsystem greift dabei auf einen zentralen Ressourcenpool zu
  - Rechenkapazität und Arbeitsspeicher
  - Datenspeicher und Backupspeicher
- Das Gesamtsystem stellt eigene Dienste, die ebenfalls verwendet werden können
  - Load-Balancer zur dynamischen Lastenverteilung auf Hintergrundsysteme
  - Datenbankdienste
  - DNS, Routing, Überwachung und sonstige Dienste

Anbieter

- Skalierbare Plattformen mit Diensten
  - Microsoft Azure
  - Amazon AWS
  - Google Cloud Computing
  - Heroku
  - u.v.m.

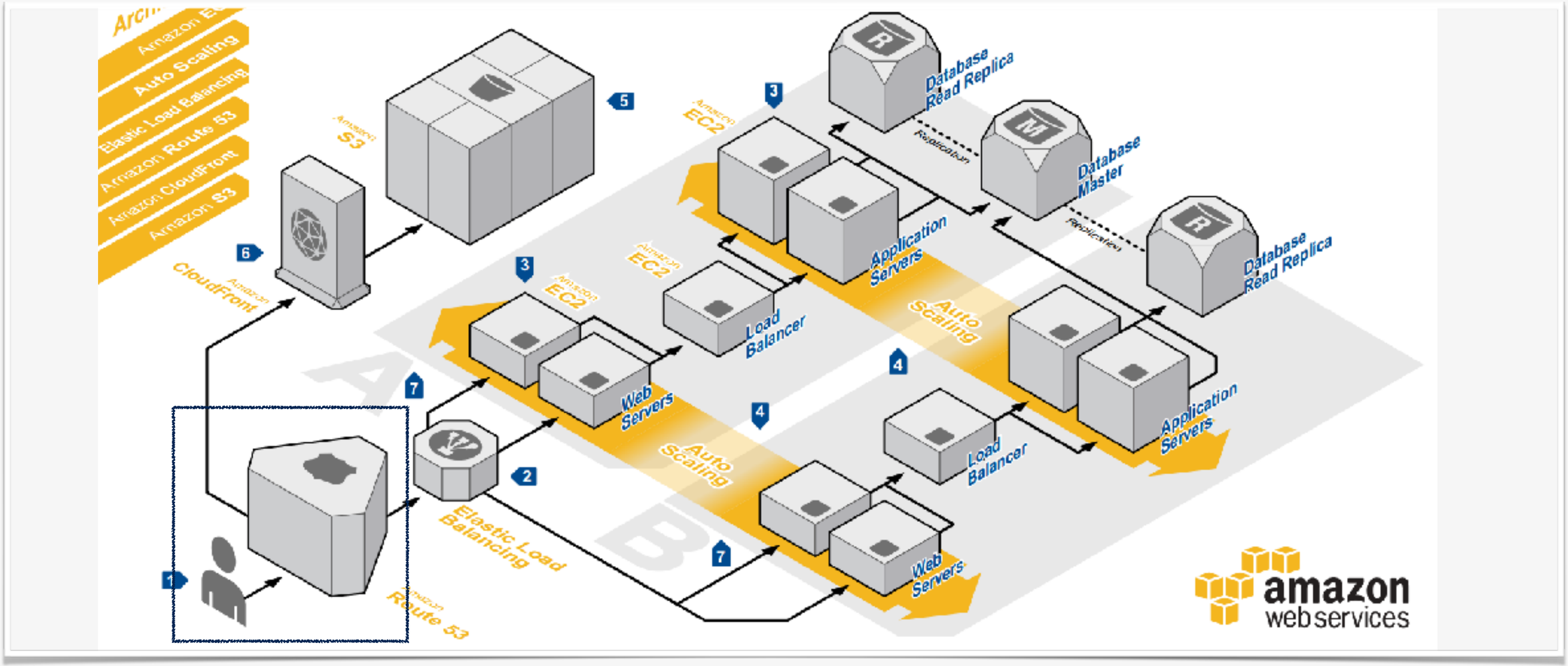
Skalierbarkeit und Dynamik

- Dienste können automatisch zu Stoßzeiten skalieren und Webdienste sind damit auch bei hoher Last immer reaktionsschnell
- Herunterfahren von Diensten und bereitgestellten Kapazitäten wenn Dienst nicht verwendet wird
- georedundante Spiegelung von Servern und Datenbanken für schnelle Zugriffszeiten

Amazon AWS - Dienstübersicht

trim/modbes

SaaS mit Cloud Infrastruktur: DNS



tr.im/amzn

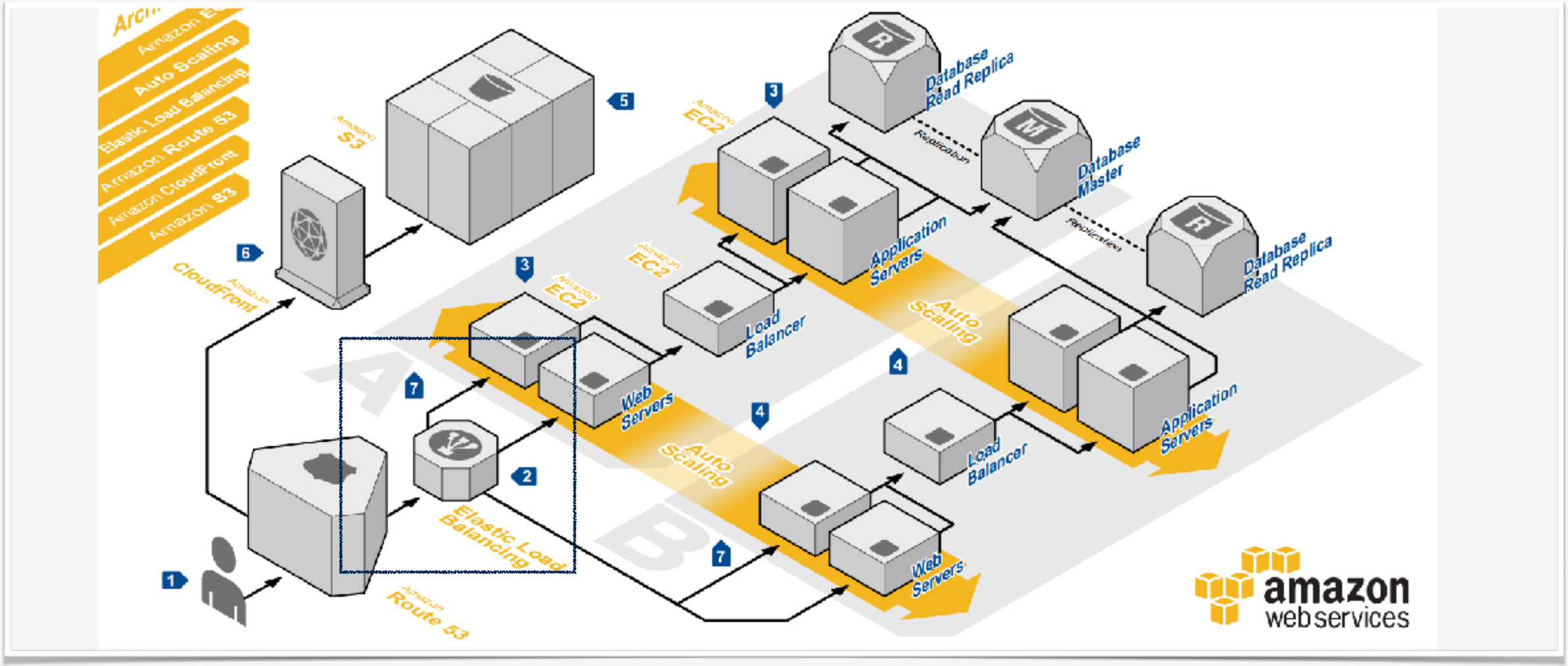
DNS Verwaltung mit Route 53

- Konfiguration von DNS
- Weiterleitung an Amazon AWS Infrastruktur
- sog. „Endpunkte“ können basierend auf Standort des Benutzers automatisch gewählt werden (latenzbasiertes DNS)
- DNS Failover
- Auswertung von Statistiken
- Überwachung und Benachrichtigung durch Amazon-Dienst (Simple Notification Service)

DNS Verwaltung mit Route 53

- Im Beispiel des SaaS
  - DNS steuert den Zugriff auf den - abhängig vom Standort - nächsten Verarbeitungsserver
  - Konfiguriert Szenarien für Failover (Routing in das nächste Rechenzentrum)
    - Schaltet ggf. Ressourcen für Kostenreduktion in dem Rechenzentrum ab

SaaS mit Cloud Infrastruktur: Load-Balancer



tr.im/amzn

Lastenverteilung mit Elastic Load Balancer

- Anfragen werden nicht an einen bestimmten Server gestellt sondern es findet ein Auswahlverfahren statt:
  - Enthält die „Resourcengruppe Webserver“ mindestens einen Server?
    - Falls nicht, wird ein Server von einer „Webserver-Vorlage“ gestartet und in die Resourcengruppe aufgenommen
  - Welcher Server hat noch Kapazität frei?
    - Weiterleitung der Anfrage an diesen Server
  - Falls kein Server eine Kapazität von X zur Verfügung stellen kann
    - Weiteren Server von einer „Webserver-Vorlage“ starten und in die Resourcengruppe aufnehmen

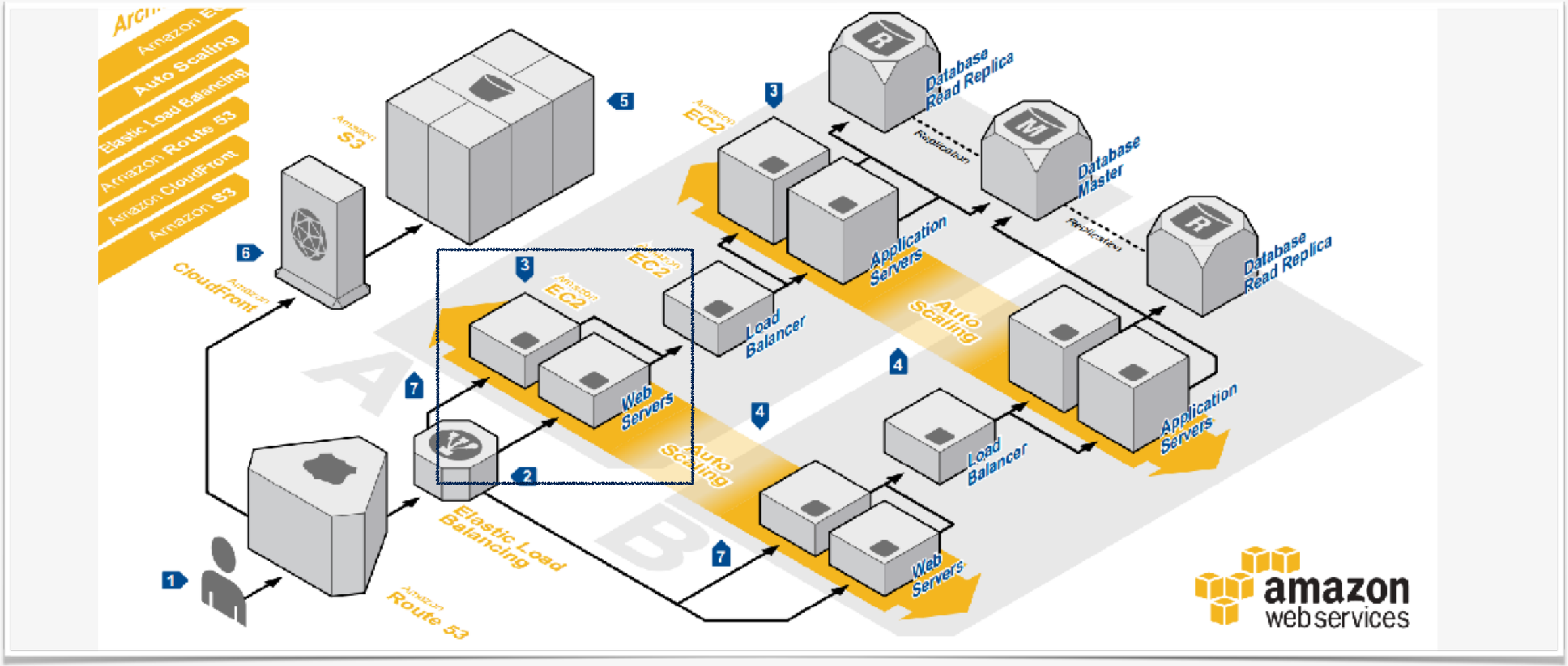
Lastenverteilung mit Elastic Load Balancer

- Der Load-Balancer kann die bereitgestellte Leistung (mögliche „Anfragen/Sekunde“) selbstständig skalieren
- Das „Auto-Scaling“ von Servergruppen stellt eine schnelle Beantwortung von Anfragen bereit
- Auto-Scaling kann auch Server herunterfahren und löschen, falls diese nicht ausreichend ausgelastet werden
  - Diese Grenzen können vom Administrator definiert werden

Elastic Load Balancer

- Im Beispiel des SaaS
  - Der Load-Balancer stellt sicher das Web-Server die Anfragen von einem Client beantwortet werden können
  - Teilt die Anfragelast zwischen vorhanden Webservern auf

SaaS mit Cloud Infrastruktur: Web-Server



tr.im/amzn

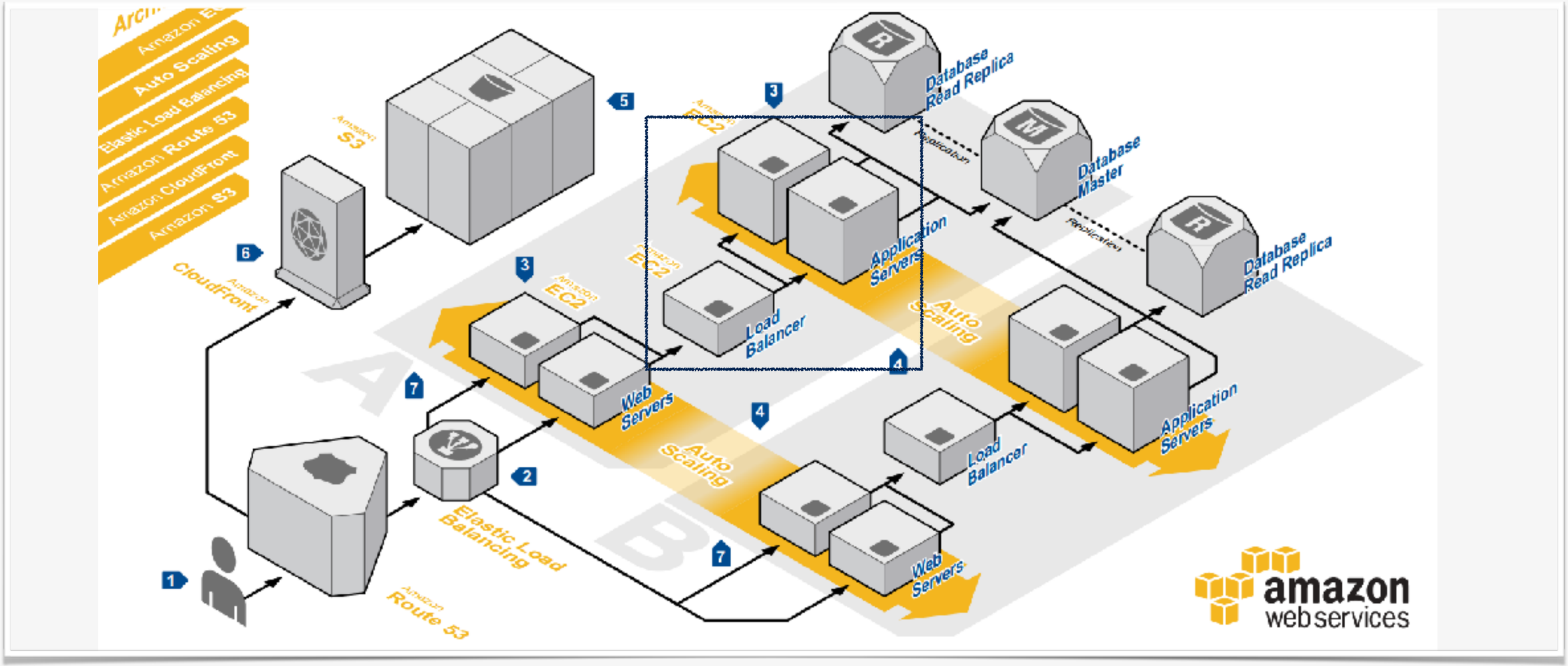
Beantwortung von Anfragen: Webserver

- Der Webserver kann Informationen von anderen Servern oder Diensten abrufen, aufbereiten und eine gewünschte Information zurückgeben
  - Statistikdaten aus einer Datenbank
  - Grafiken von der Festplatte
  - Layout und Design
  - Ausführung von dynamischen Quellcode (bspw. PHP)
- Kombination der Ergebnisse wird als Webseite zurückgegeben

Webserver

- Im Beispiel des SaaS
  - Wickelt die Authentifizierung des Anfragenden ab
  - Legt einen neuen Arbeitsauftrag an
  - Speichert Informationen zu dem Vorgang in der Datenbank und Dateisystem ab
    - Keine Speicherung auf dem Webserver, da nicht garantiert werden kann, dass dieser Server die nächste Anfrage übernimmt!

SaaS mit Cloud Infrastruktur: Application-Server



tr.im/amzn

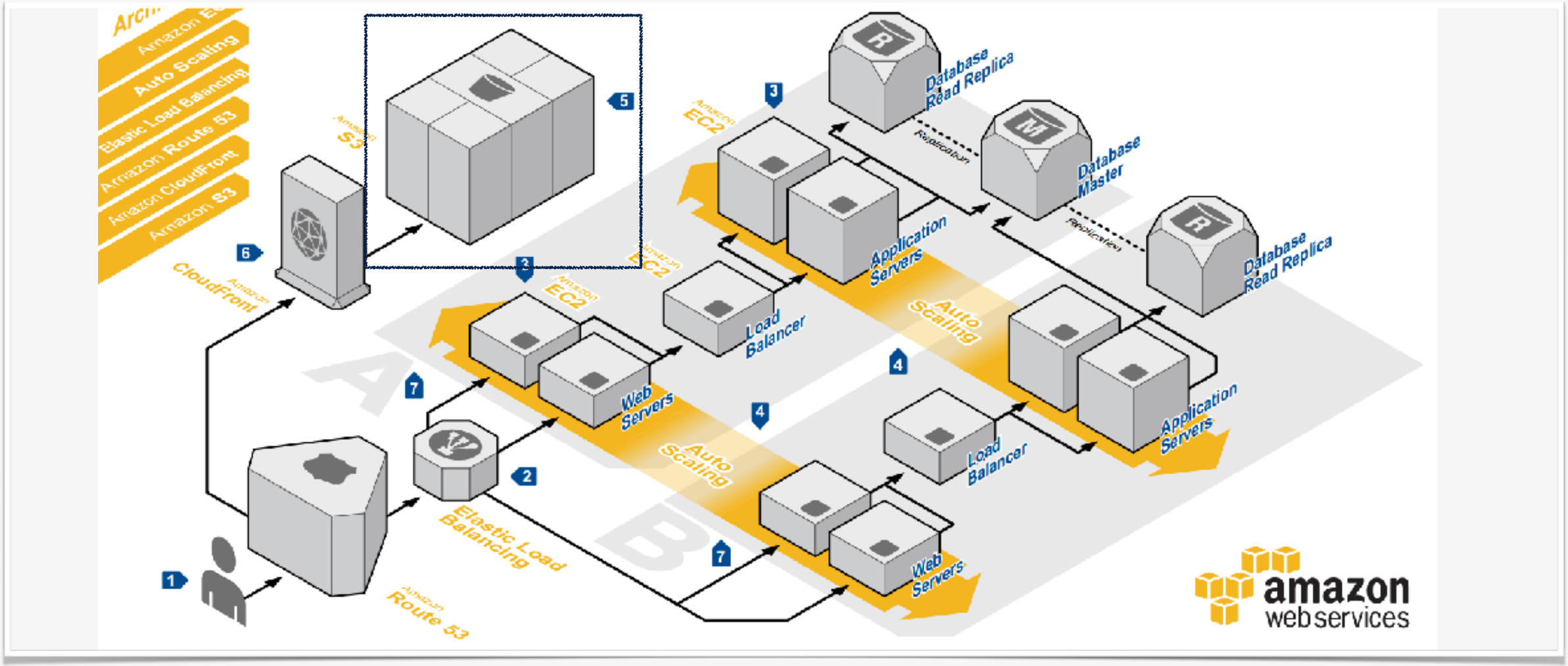
Abarbeitung der Anfrage

- Der Webserver kommuniziert mit der hinterliegenden Infrastruktur
  - Datenbankserver (Daten und Cache-Datenbanken)
  - Anwendungsserver
  - Cache
  - Datenspeicher

Anwendungsserver

- Im Beispiel des SaaS
  - Vorgeschalteter Load-Balancer sorgt für genügend Kapazität
  - Datenbank (Amazon RDS-Dienst) speichert Arbeitsauftrag
  - PDF-Datei für die Verarbeitung wird in Amazon S3 „Bucket“ gespeichert (Festplattenspeicher)
  - Für den Verarbeitungsprozess der Datei wird ein Prozess auf einem separaten Server gestartet
    - Sollten weitere Ressourcen notwendig sein, werden weitere Server bereitgestellt

SaaS mit Cloud Infrastruktur: S3 Speicher



tr.im/amzn

Speicherung der Daten im S3-Bucket

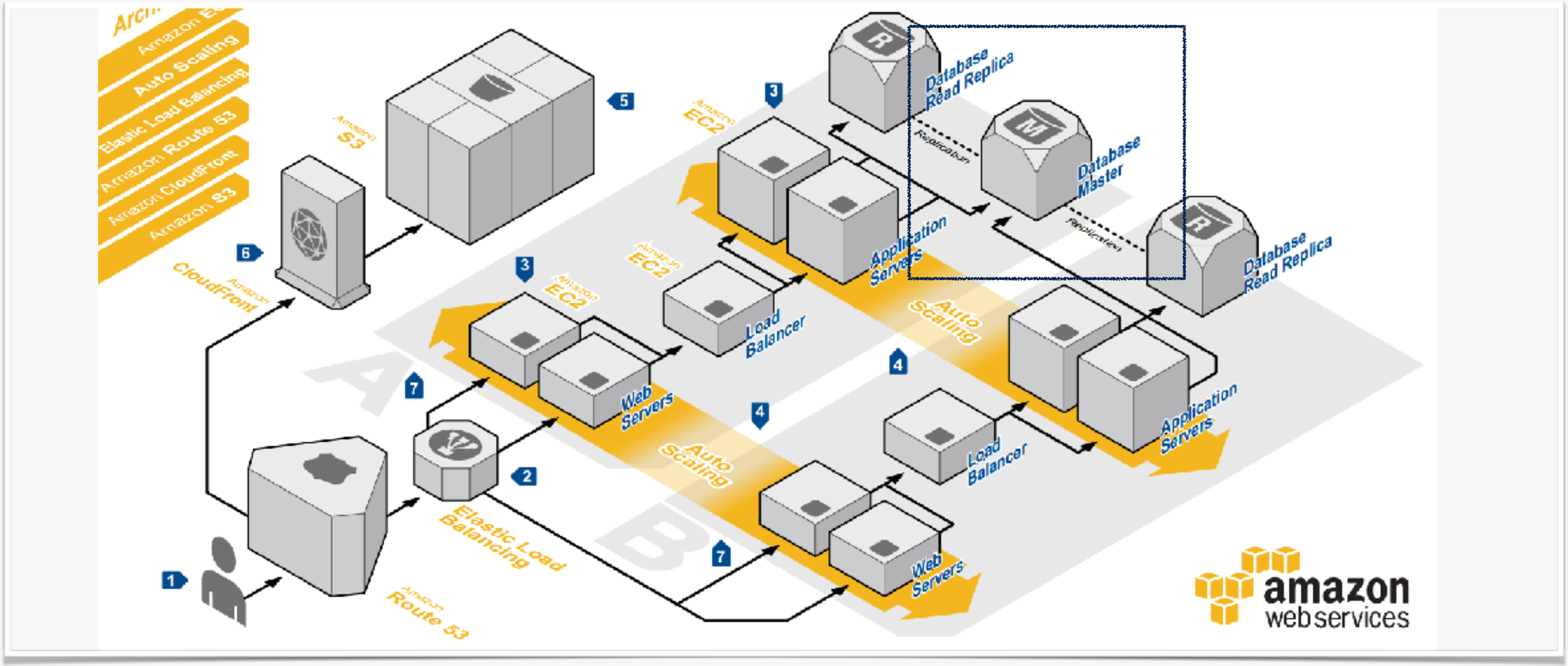
- Der S3-Bucket ist ein Datenspeicher für Dateien bis zu einer Dateigröße von 5 TB
- Alle Amazon-Resourcen können auf einen S3-Bucket und dessen Inhalt zugreifen
- Es ist der zentrale Speicherort für Logs, Anwendungsdateien, Konfigurationsdateien
- Die Daten in einem Bucket müssen manuell gelöscht werden

  (Server haben einen temporären Speicher)

S3-Speicher

- Im Beispiel des SaaS
  - Beinhalten Log-Dateien, Konfigurationsdateien und die „Eingabedatei“
  - Server holen sich die Ressourcen und legen das Ergebnis im Bucket ab

SaaS mit Cloud Infrastruktur: Webserver



tr.im/amzn

Aktualisierung der Informationen in der Datenbank

- Amazon AWS bietet NoSQL-Datenbanken und relationale Datenbanksysteme an
- Die Lese- und Schreibgeschwindigkeit kann für jede Tabelle granular definiert werden
- Datenbanken können per Knopfdruck zwischen verschiedenen Amazon Rechenzentren synchronisiert werden

Datenbanken

- Im Beispiel des SaaS
  - Aktualisieren des Kundenprofils innerhalb der Datenbank
  - Speicherung der Metriken
    - Wie lange wurde die Datei verarbeitet?
    - Aktualisierung des Kundenkontos (eCommerce)

Bereitstellung der Datei

- Im Beispiel des SaaS
  - Datei wird nach der Verarbeitung in einem gesonderten S3-Bucket abgelegt und kann von dem Client heruntergeladen werden

Zusätzliche AWS-Dienste

- CloudFront
  - Content Distribution Network
  - statische Websitedaten (Grafiken, CSS/Javascriptdateien) einem Client besonders schnell zum Abruf anbieten
  - Spezielle Caching-Optionen
  - Im SaaS-Beispiel: Bilder, CSS und Javascriptdateien der Website speichern

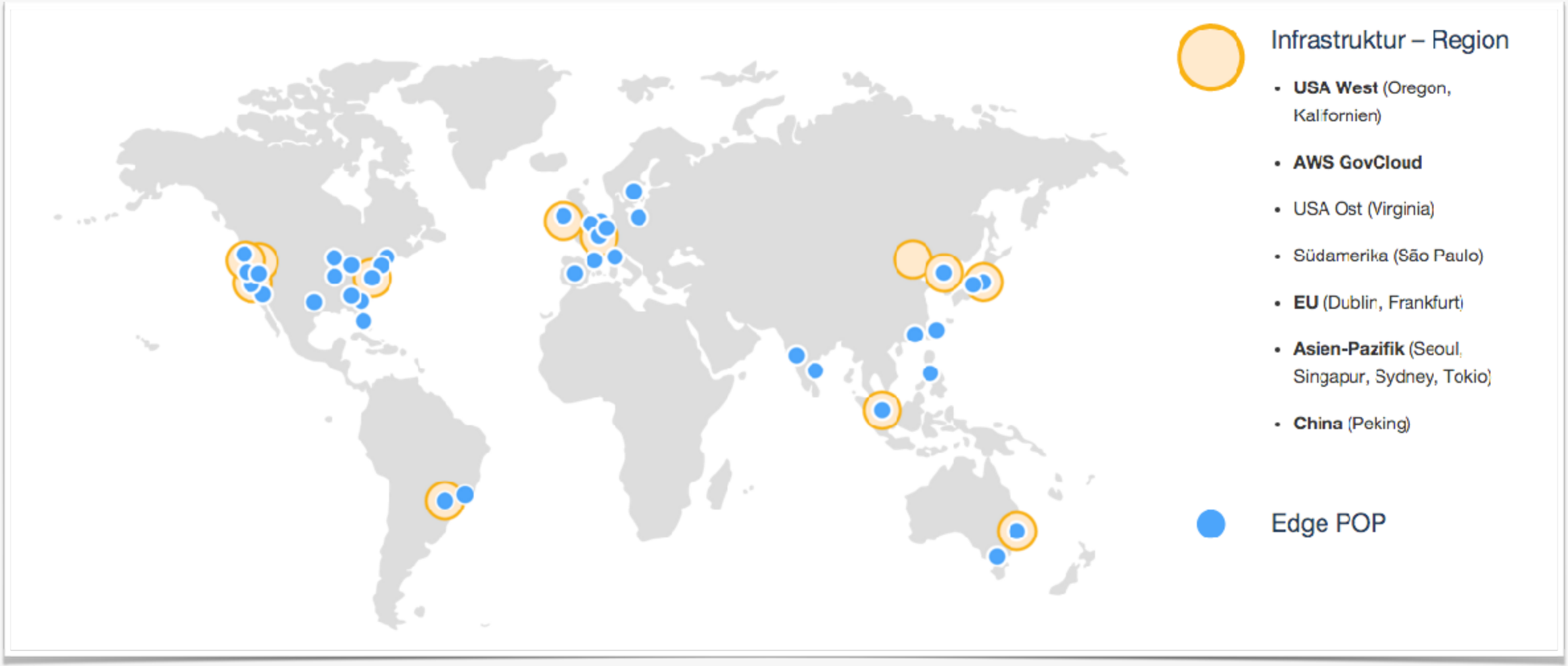
Zusätzliche AWS-Dienste

- Elastic Beanstalk
  - AWS Infrastruktur basierend für ein Quellcode-Repository bereitstellen
  - Live-Umgebung und Staging-Umgebung für Tests
  - Migration von Staging-Umgebung mit einem Klick in Produktivumgebung

Zusätzliche AWS-Dienste

- Glacier
  - preisweiter Langzeit Archivspeicher
  - Kosten für Restore: Je nach Zeitraum der Wiederherstellung
    - wenige Stunden bis hin zu mehreren Tagen Wartezeit

Amazon AWS Datenzentren



trim/amzn

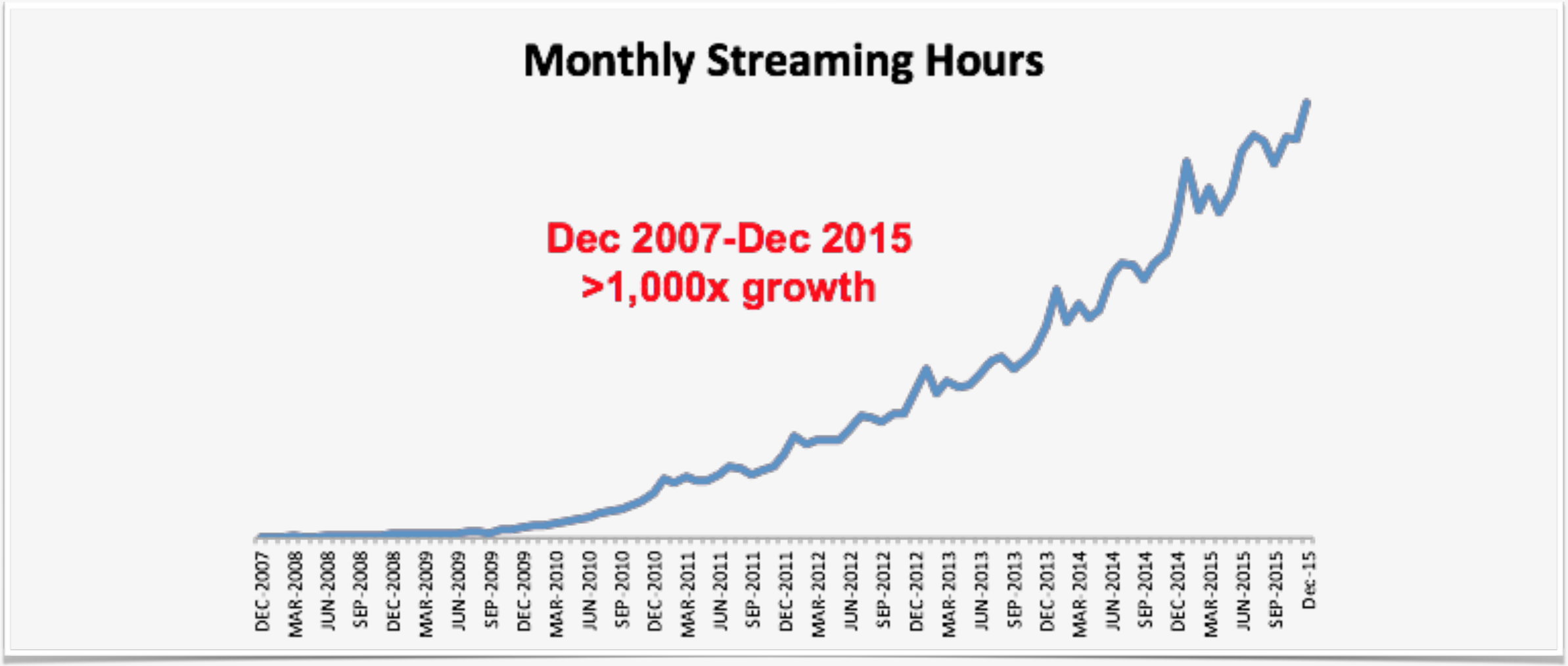
Vorteile

- Hohe Ausfallsicherheit
- Keine Vorhaltung von redundanter Hardware
- Keine Vorhaltung von georedundanter Hardware
- Produkt kann „klein beginnen“ und skaliert mit der Anwenderzahl mit
- Umsetzung von „Big-Data“ Projekten

Nachteile

- komplexes Abrechnungsmodell
  (Einheiten, Stunden, Datentransfer, Grundgebühren u.v.m.)
- Projekt muss für Cloud-Dienste optimiert werden
  - horizontale Skalierbarkeit der Aufgabe (Aufgabe muss in Teilprobleme aufgeteilt werden können)
- Wechsel zu einer anderen Plattform mit sehr hohem Aufwand verbunden
  - Nutzung von anbieterspezifischen Diensten
    (bspw. Amazon eigene Datenbank „DynamoDB“)
  - Migration und Tests

Skalierung der Dienste am Beispiel Netflix



tr.im/**netflix**

Netflix

- setzt auf die Amazon AWS Infrastruktur für die Datendienste
- führt dauerhaft in der Produktivumgebung Tests durch
  - eigener Bot „Chaos-Monkey“ zerstört rund um die Uhr Produktivserver
  - pro Jahr um die 65.000 Serverinstanzen während des normalen Streamingbetriebs zerstört und der Failoverbetrieb getestet
  - Andere Monkeys testen
    - Latenzen
    - Sicherheit
    - Aufräumarbeiten (Clean-Ups)
    - vieles mehr

Caching mit Redis

Ergebnisse zwischenspeichern

Cache

- Ein Cache speichert Informationen für einen schnelleren (wiederholten) Zugriff
- Diese Informationen können vorab dynamisch generiert worden sein
  - Aufbereitete Informationen aus einer Datenbank
  - Ausgabe eines Interpreters einer dynamischen Skriptsprache (bspw. PHP)
  - Teilinhalte einer Website die sich nur gelegentlich ändern (bspw. Infoseite)
- Im Webbereich sind Caches üblicherweise
  - dateibasiert (Ablage der Datei auf der Festplatte)
  - datenbankbasiert (NoSQL Datenbank)
  - „In Memory“ gecached
  - oder ein hybrid: „In-Memory Datenbank“

Redis

- Ist eine open source In-Memory Datenbank
- Basiert auf einem „Key-Value-Store“, wie bspw. Hashtabellen (nicht relational)
- bietet sich für einfache Datenstrukturen an
  - Ein beliebiger Wert wird unter Angabe eines Schlüssels eingetragen
  - Dieser Wert kann unter Angabe des Schlüssels wieder abgerufen werden
- Sehr performant
  - bis zu 80.000 Lesevorgänge pro Sekunde
  - bis zu 100.000 Schreibvorgänge pro Sekunde auf herkömmlicher Hardware

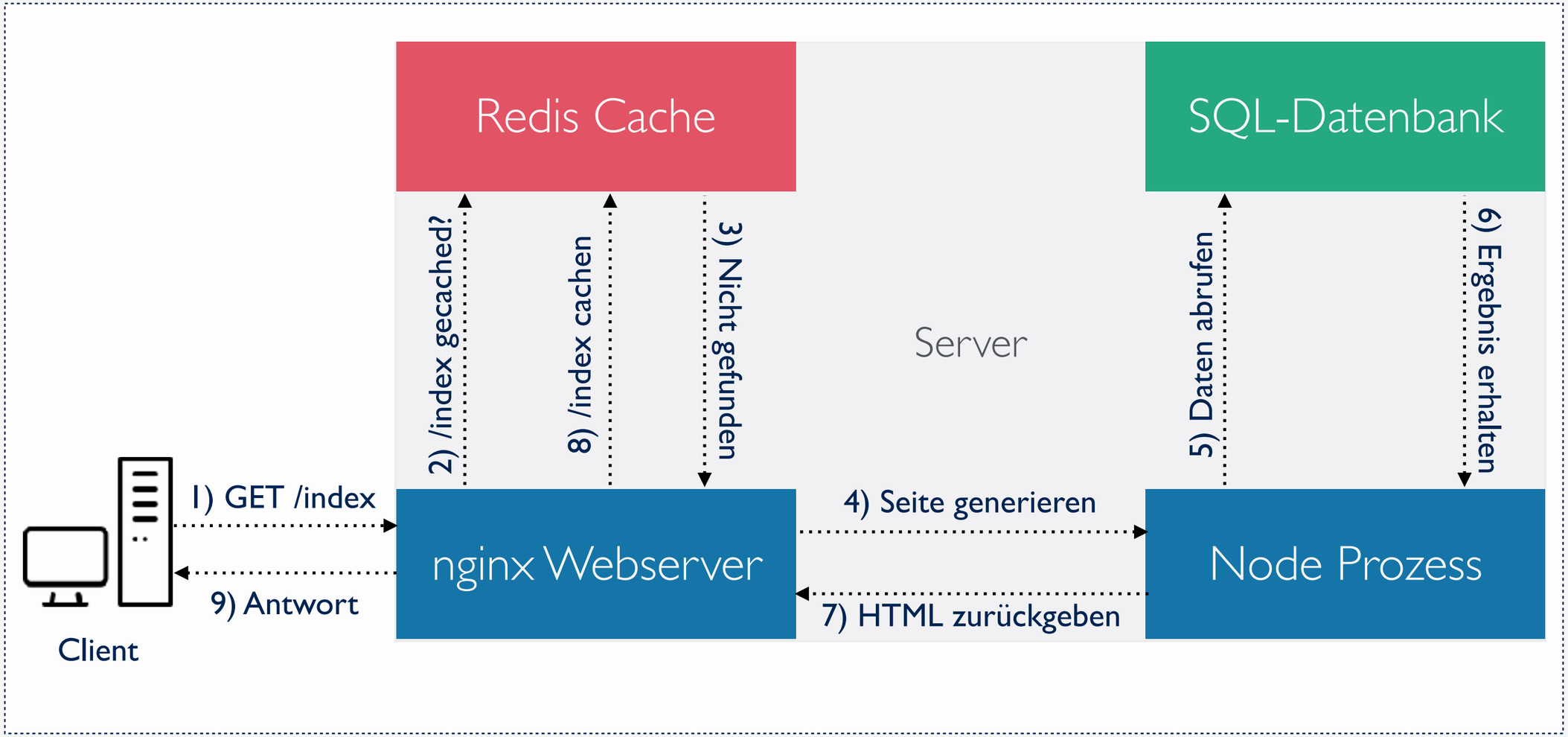
Wo kann gecached werden?

- Nginx kann als Reverse Proxy betrieben werden und speichert so Webinhalte zwischen, bzw. ruft diese von internen Systemen ab
- Auch Datenbankabfragen können mit Redis zwischengespeichert werden, bspw.
  - Komplexe Auswertungen
  - Abfragen die eine hohe Auslastung des Servers bezwecken
  - Abfragen von Werten, die sich nicht verändern (aggregierte Umsatzzahlen der Vorwoche)

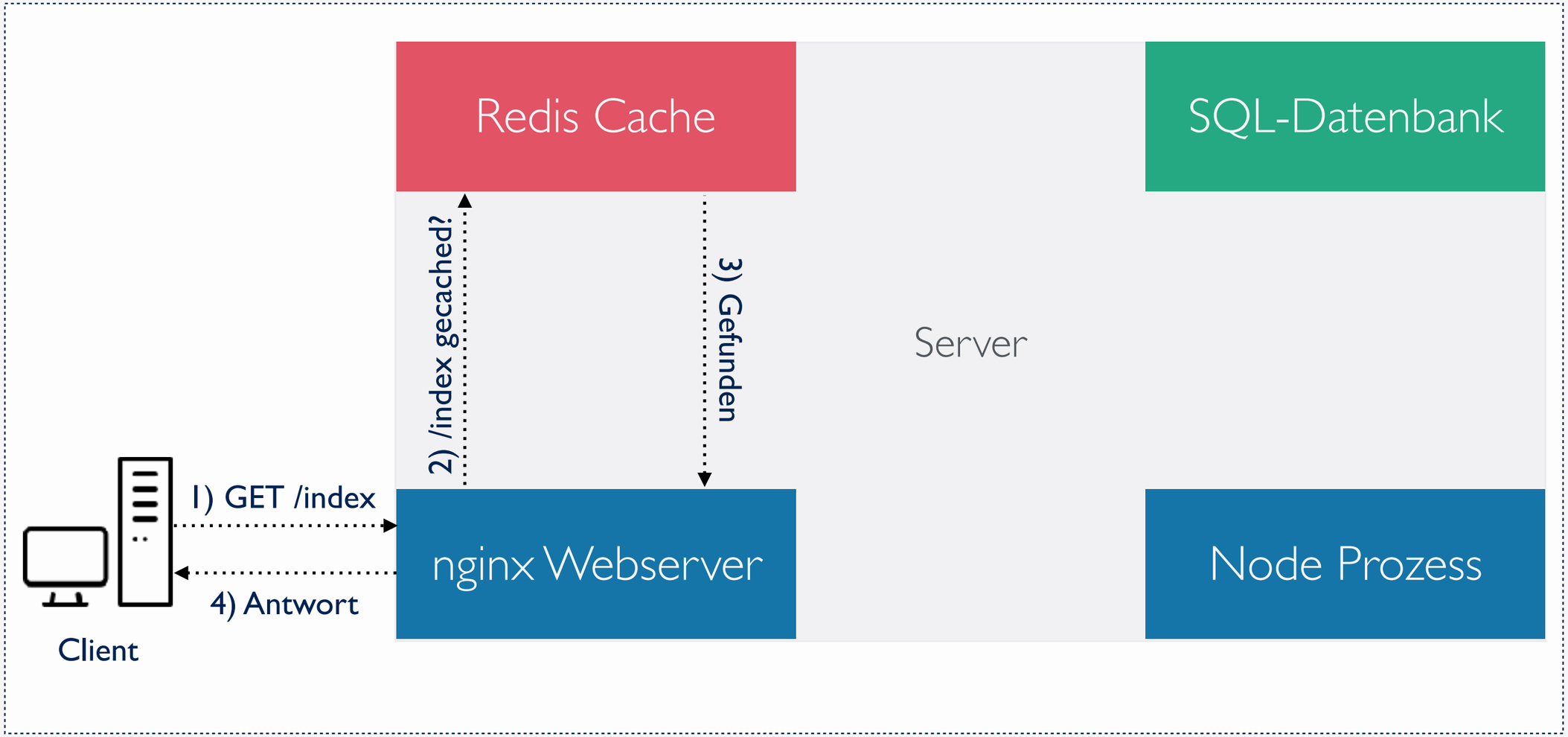
Cache Gültigkeit

- Die Dauer der Gültigkeit muss durch den Entwickler festgelegt werden
  - Definierte Zeitspanne
  - Neustart des Systems (ggf. Verlust des Caches)
  - Löschen des Caches nach Aktualisierung der Daten durch externen Prozess
  - Manuelles Löschen des Caches

Cachen mit Redis - keine Daten im Cache



Cachen mit Redis - Daten im Cache



Vorteile des Caching

- Anfragen an den Webserver werden schneller beantwortet
- Jede zwischengespeicherte Information muss nicht generiert werden, was die Auslastung des Servers vermindert
- pro Webserver können mehr Anfragen beantwortet werden
- Bei rechenintensiven Datenbankabfragen
  - weniger Last auf dem Datenbankserver
  - schnellere Antwortzeiten bei anderen parallelen Abfragen

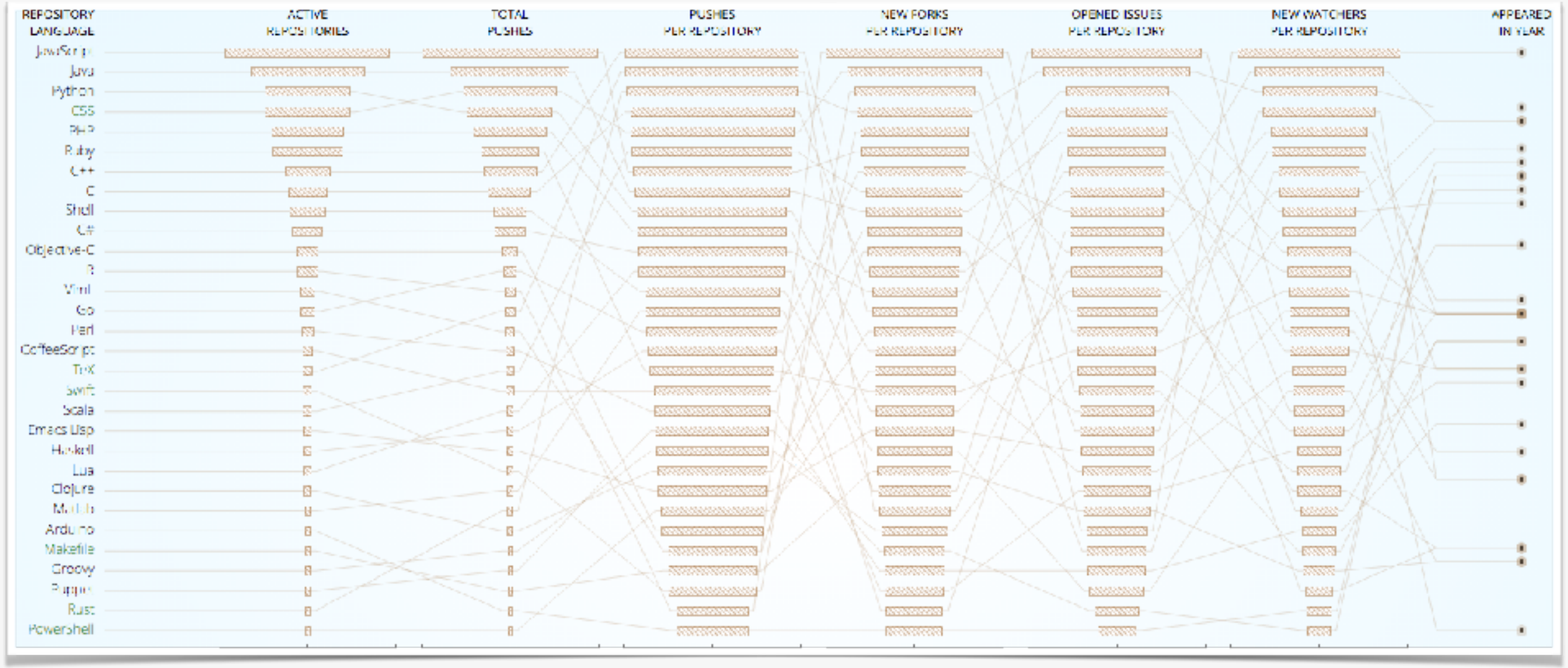
Nachteile des Caching

- Die Gültigkeit des Caches muss gewartet werden
  - Informationen die nicht mehr Up2Date sind, müssen entfernt oder aktualisiert werden
- Kein Cachen bei der Verwendung von Cookies möglich
- Nicht alle Bereiche einer Webseite sollten gecached werden
  - Echtzeitstatistiken
  - Interne Bereiche einer Webseite

Übersicht der Websprachen

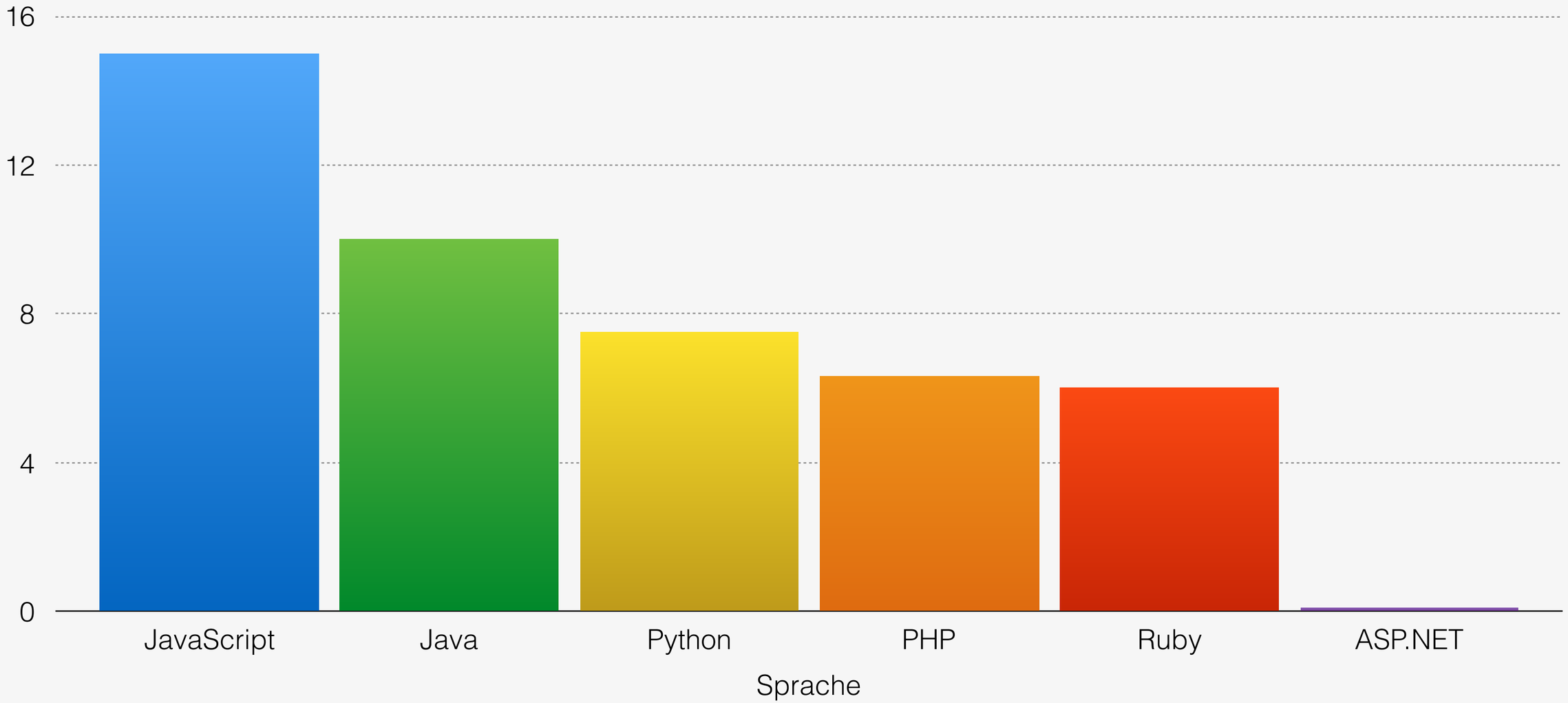
Aktuelle Sprachen und deren Verteilung

Übersicht über aktuelle GitHub Repositories



trim/githut

Aktuelle Verteilung



JavaScript

- Client oder serverbasiert (bspw. jQuery / Node.JS)
- Plattformunabhängig
  - Windows, Mac, div. Linux-Distributionen, Embedded & Internet of Things

```
// Kommentartext

function eineFunktion() {
    var autoName = "BMW";
}
```

/dpetrasch/DHBW

Java

- Java Web-Applications (J2EE)
- Plattformunabhängig
  - Windows, Mac, div. Linux-Distributionen

```
public void startDocument()
throws SAXException
{
  nl();
  nl();
  emit("START DOCUMENT");
  nl();
  emit("<?xml version='1.0' encoding='UTF-8'?>");
  nl();
}
```

/dpetrasch/DHBW

Python

- Plattformunabhängig
  - Windows, Mac, div. Linux-Distributionen

```
   # Python 3: Fibonacci series up to n
>>> def fib(n):
>>>     a, b = 0, 1
>>>     while a < n:
>>>         print(a, end=' ')
>>>         a, b = b, a+b
>>>     print()
>>> fib(1000)
```

/dpetrasch/DHBW

PHP

- In Version 5 auch objektorientiert (v4 nur prozedural)
- Plattformunabhängig
  - Windows, Mac, div. Linux-Distributionen

```php
public function execute($handlers, $path, $method, $data='', $headers=array()) {
    $config = $this->apiContext->getConfig();
    $httpConfig = new PPHttpConfig(null, $method);
    $httpConfig->setHeaders($headers +
        array(
            'Content-Type' => 'application/json'
        )
    );

    return $response;
}
```

/dpetrasch/DHBW

Ruby

- Modular mit „Gems“
- Bekannte MVC-Bibliothek ist „Ruby on Rails“
- Online Beispiel auf tryruby.org
- Plattformunabhängig
  - Windows, Mac, div. Linux-Distributionen

```ruby
def welcome(name)
    puts "howdy #{name}"   # inside double quotes, #{ } will evaluate the variable
end

welcome("nana")           # traditional parens
```

/dpetrasch/DHBW

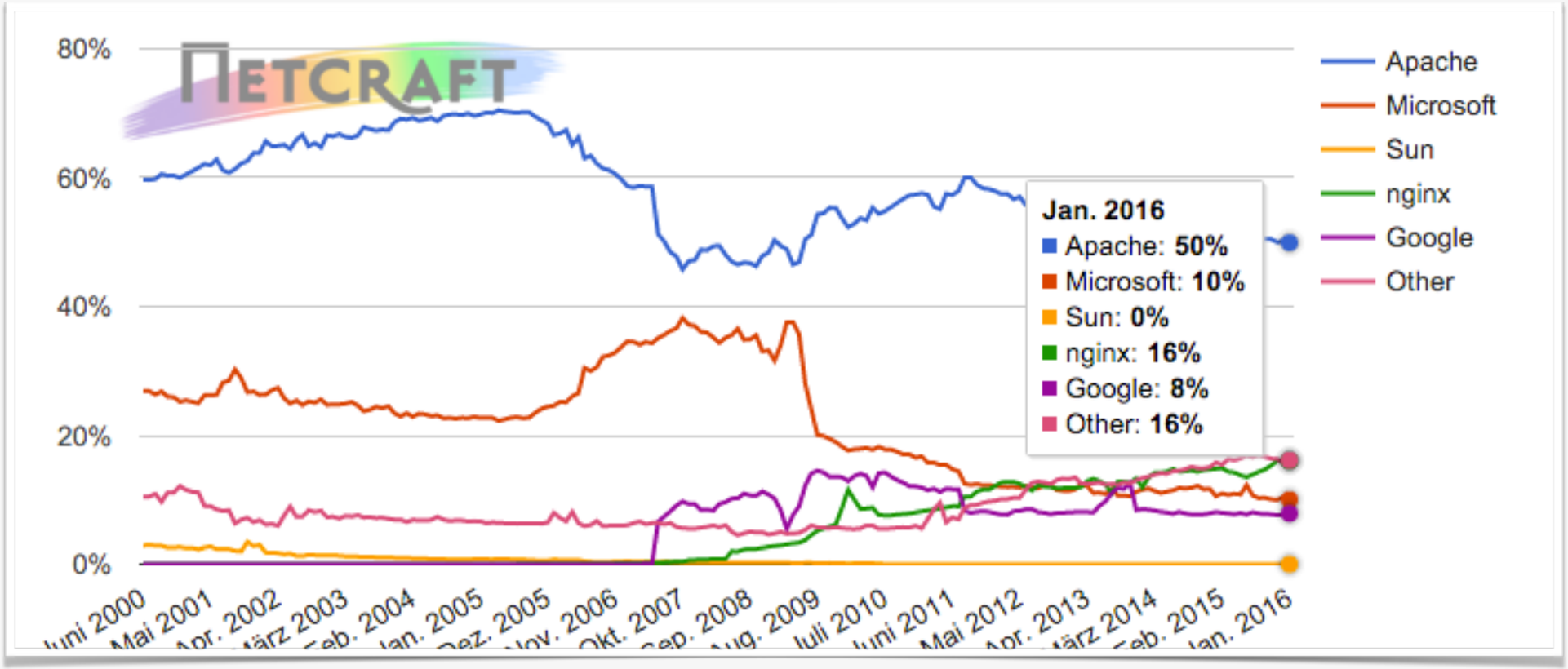
ASP.NET

- Modular mit „NuGet Packages“
- Plattformabhängig
  - Windows, andere Plattformen mit Mono

```
public class IntroParams
{
   [Params(100, 200)]
   public int A { get; set; }

   [Params(10, 20)]
   public int B { get; set; }

   public void Benchmark()
   {
     Thread.Sleep(A + B + 5);
   }
 }
```

/dpetrasch/DHBW

Webserver Marktanteile

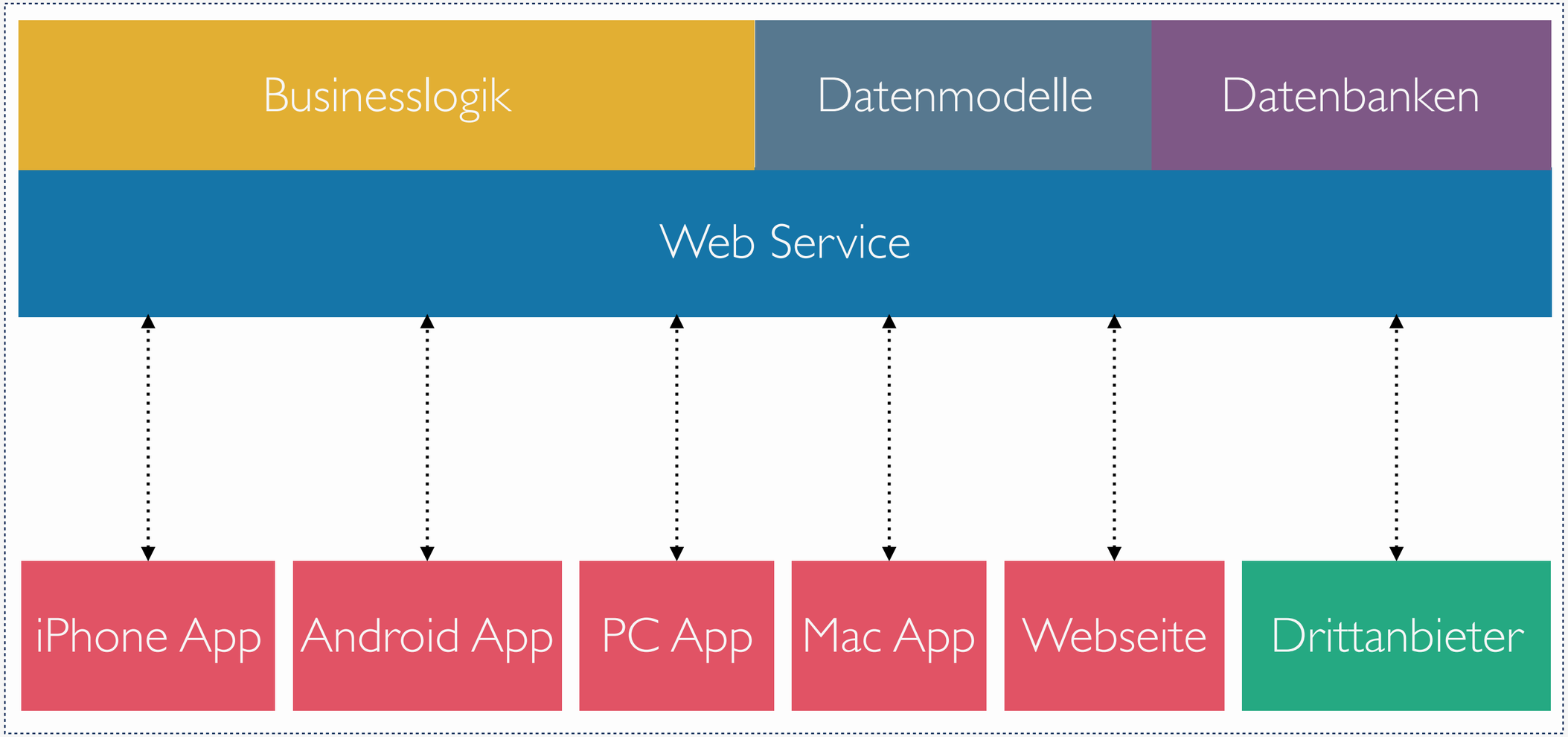


tr.im/netcraft

Web Services

REST und SOAP

Anwendungsgebiet: Web Services



Was sind Web Services?

- eine im Internet veröffentlichte Software die über Standardschnittstellen angesprochen werden kann
- Interaktion zwischen Client und Server geschieht durch den Austausch von XML-basierten Nachrichten über Internetprotokolle
- die Technologien hinter Web Services sind plattformunabhängig
- mit Web Services können eigene Dienste bereitgestellt werden, die Informationen für Webseiten, Apps und Anwendungen gleichzeitig bereitstellen

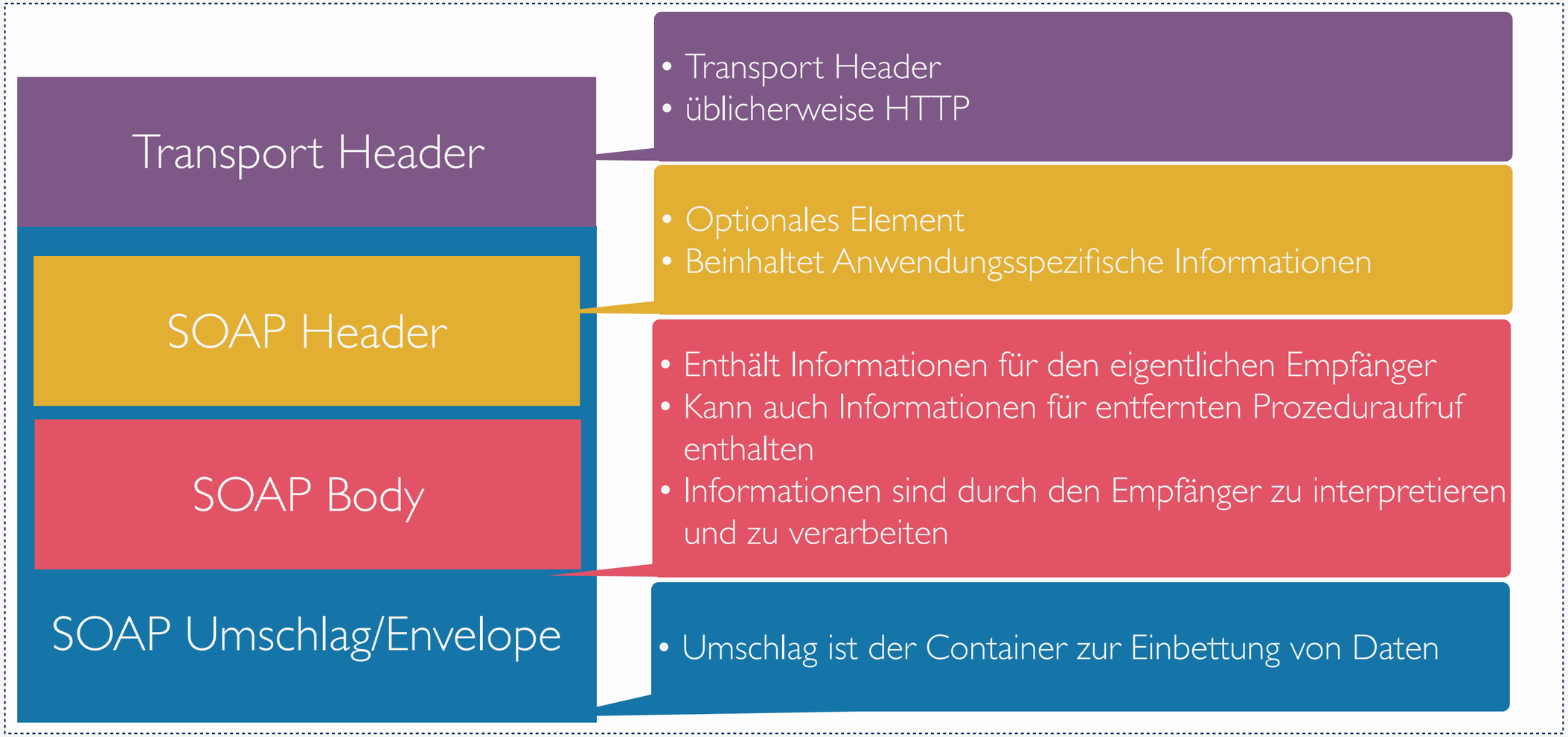
SOAP: Allgemeines

- SOAP
  - steht für Simple Object Access Protocol
  - ist ein Netzwerkprotokoll mit dessen Hilfe Daten zwischen Systemen ausgetauscht werden können oder entfernte Prozeduren aufgerufen werden können
  - ist ein industrieller Standard des World Wide Web Consortiums (W3C)
- Basiert auf XML
- Plattformunabhängig
- Programmiersprachenunabhängig
- Es existiert kein bestimmtes Transportprotokoll, üblicherweise wird HTTP benutzt

SOAP: Anwendungsoptionen

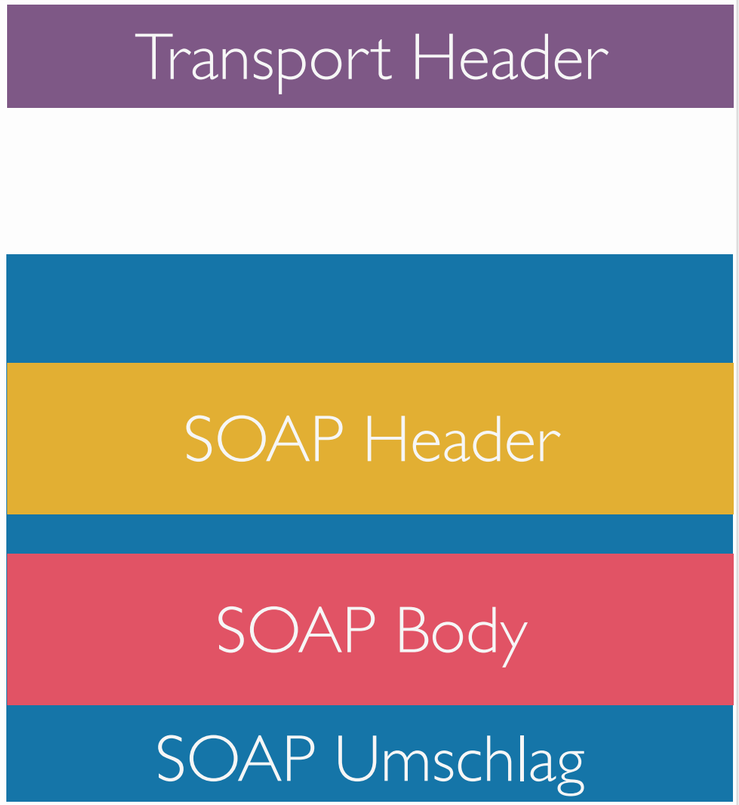
- RPC (Remote Procedure Call)
  - Es wird eine Methode des Servers per Web Service aufgerufen
  - Funktionsname und Namen werden in der SOAP-Nachricht an den Server übertragen
- Dokumentbasiert
  - Es werden strukturierte Informationen versendet
  - Der Webserver verarbeitet diese Informationen und sendet das Ergebnis sofort oder zeitlich versetzt zum Server zurück

Aufbau einer SOAP-Nachricht



Struktur von SOAP-Nachrichten

```
POST /soapnachrichtenendpunkt HTTP/1.1
Content-Type: application/soap


<?xml version="1.0"?>
<s:Envelope xmlns:s=„http://www.w3.org/2003/05/soap-envelope">

    <s:Header>
      . . .
    </s:Header>

    <s:Body>
      . . .
    </s:Body>

</s:Envelope>
```



/dpetrasch/DHBW

SOAP-Envelope

- Grundgerüst einer SOAP-Nachricht
- Ist ein Behälter vergleichbar mit einem Briefumschlag
- In dem SOAP-Envelope ist der SOAP-Header und der SOAP-Body enthalten

```
<?xml version="1.0"?>
<s:Envelope xmlns:s=„http://www.w3.org/2003/05/soap-envelope“>

 <!-- optionaler SOAP-Header -->
 <!-- SOAP-Body -->

</s:Envelope>
```

/dpetrasch/DHBW

SOAP-Header

- Optionaler Bestandteil
- falls vorhanden, muss der SOAP-Header vor dem Body stehen
- inhaltsunabhängige Daten und anwendungsspezifische Angaben

```
<x:ersterBlock xmlns:x=„http://example.com“>
 <!-- Variablen -->
</x:ersterBlock>
```

/dpetrasch/DHBW

SOAP-Body

- Beinhaltet die eigentliche Nachricht
- Keine festgelegte Struktur
- Muss im XML-Format geschrieben werden
- Bei RPC enthält er Informationen zum Funktionsaufruf
  - Methodenname
  - Parameterliste
  - Rückgabewert
- Optional: Attachment
  - Keine Beschränkung für das Dateiformat
  - Zum Anhängen von Dokumenten

SOAP-Fault

- Wird verwendet um Informationen über aufgetretene Fehler zu übermitteln
- SOAP-Fault ist ein Unterelement des Bodys
- Besteht aus den Unterelementen
  - Code: Fehlerklassifizierung
  - Reason: dient der menschenlesbaren Fehlerbeschreibung
  - Optional: Beschreibung wo genau der Fehler aufgetreten ist:

    Node, Role und Detail

RPC-Nachricht

```
<?xml version='1.0' encoding="UTF-8?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
        <m:reservation xmlns:m="http://travelcompany.org/reservation"
          env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
          env:mustUnderstand="true">
          <m:reference>uuid:093a2da1-q345-739r-ba5d</m:reference>
          <m:date>2004-05-10</m:date>
        </m:reservation>
    </env:Header>
    <env:Body>
        <p:itinerary xmlns:p="http://travelcompany.org/reservation/travel">
          <p:setReservation>
              <p:departing>Duesseldorf</p:departing>
              <p:arriving>Muenchen</p:arriving>
              <p:departureDate>2004-05-29</p:departureDate>
              <p:departureTime>mid-morning</p:departureTime>
              <p:class>business</p:class>
          </p:setReservation>
        </p:itinerary>
    </env:Body>
</env:Envelope>
```

Endpunkt für den Webservice

Funktionsaufruf der Funktion „setReservation“

Parameter:
Departing, Arriving, departureDate, departureTime, class

/dpetrasch/DHBW

Dokumentbasierte Nachricht

```
<?xml version='1.0' encoding="UTF-8?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
        <m:reservation xmlns:m="http://travelcompany.org/reservation"
          env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
          env:mustUnderstand="true">
          <m:reference>uuid:093a2da1-q345-739r-ba5d</m:reference>
          <m:date>2004-05-10</m:date>
        </m:reservation>
    </env:Header>
    <env:Body>
        <p:itinerary xmlns:p="http://travelcompany.org/reservation/travel">
            <departing>Duesseldorf</departing>
            <arriving>Muenchen</arriving>
            <departureDate>2004-05-29</departureDate>
            <departureTime>mid-morning</departureTime>
            <class>business</class>
        </p:itinerary>
    </env:Body>
</env:Envelope>
```

Endpunkt für den Webservice

Es werden die Variablen departing, arriving, departureDate, departureTime und class übergeben

Der Empfänger muss selbstständig die Informationen extrahieren!

/dpetrasch/DHBW

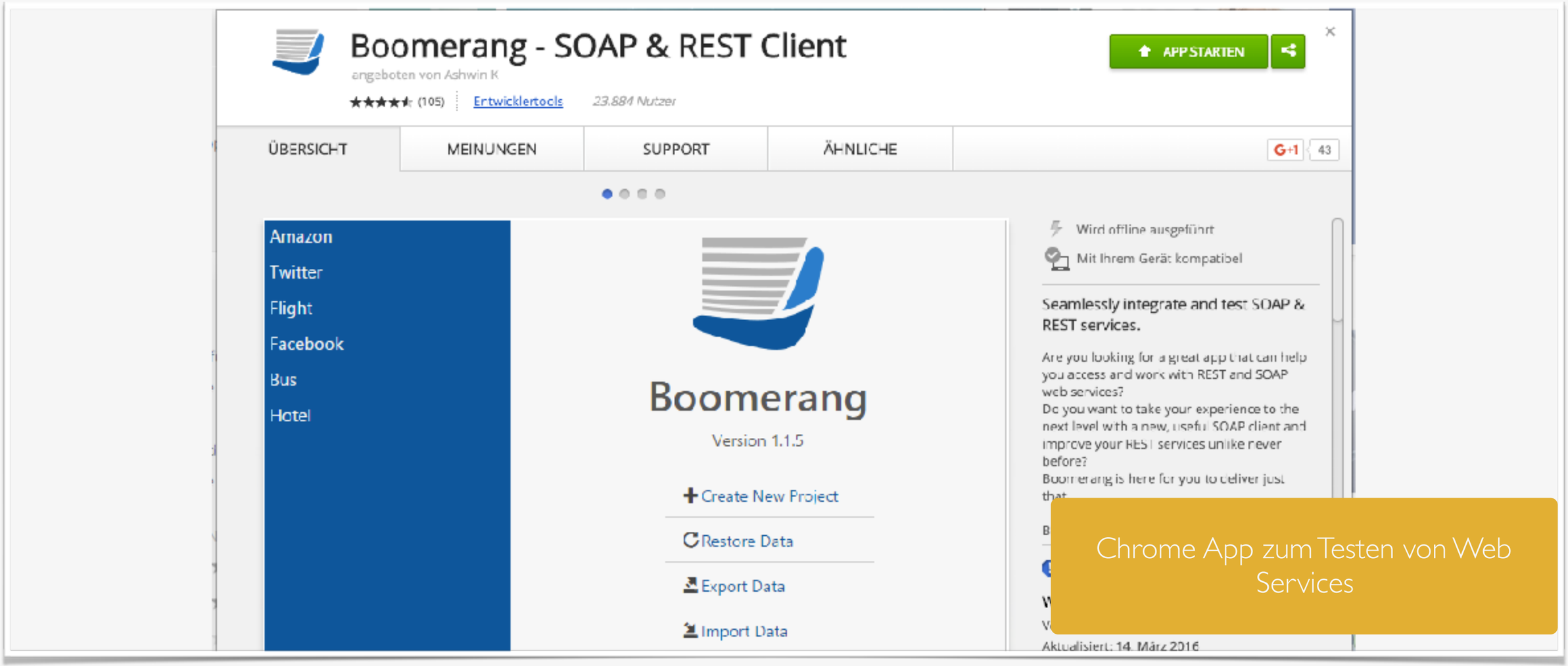
SOAP: Unterstütze Datentypen

- Es können einfache und zusammengesetzte Datentypen verwendet werden
  - Einfache
    - Byte, Integer, Long (Zahlen)
    - Float, Double (Gleitkommazahlen)
    - Char und String (Buchstabe und Text)
    - Boolean (Ja/Nein)
  - Zusammengesetzte Datentypen
    - Array
    - Strukturen (Klassen)

Verwendung von Onlinediensten: Wetterdienst

- Verschiedene Dienste stellen per SOAP Daten für den Abruf bereit
  - Wetterdienste
  - Aktieninformationen
  - Abruf von Bankleitzahlen
- Nutzen einer SOAP-Schnittstelle am Beispiel eines Wetterdienstes
- Graphische Tools oder Webseiten vereinfachen Tests des Dienstes

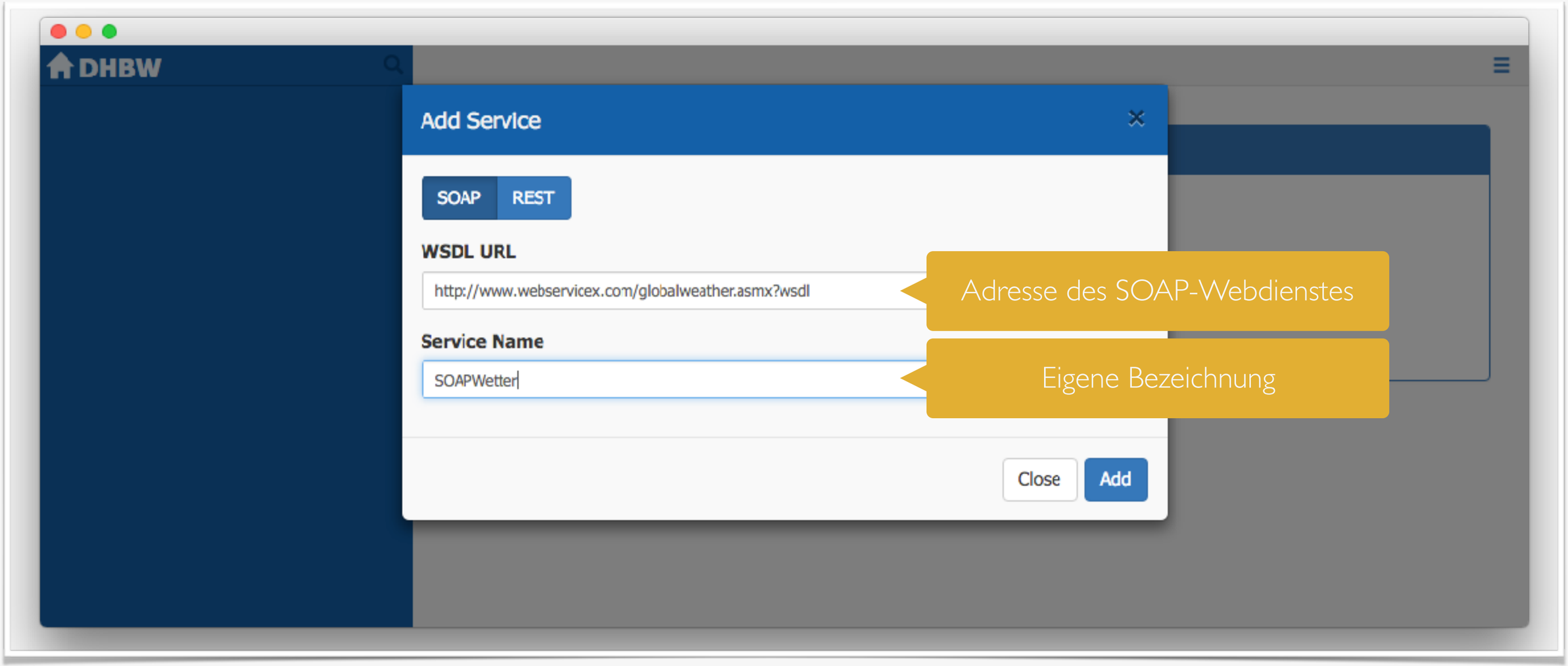
Boomerang Test-Client



Chrome App zum Testen von Web Services

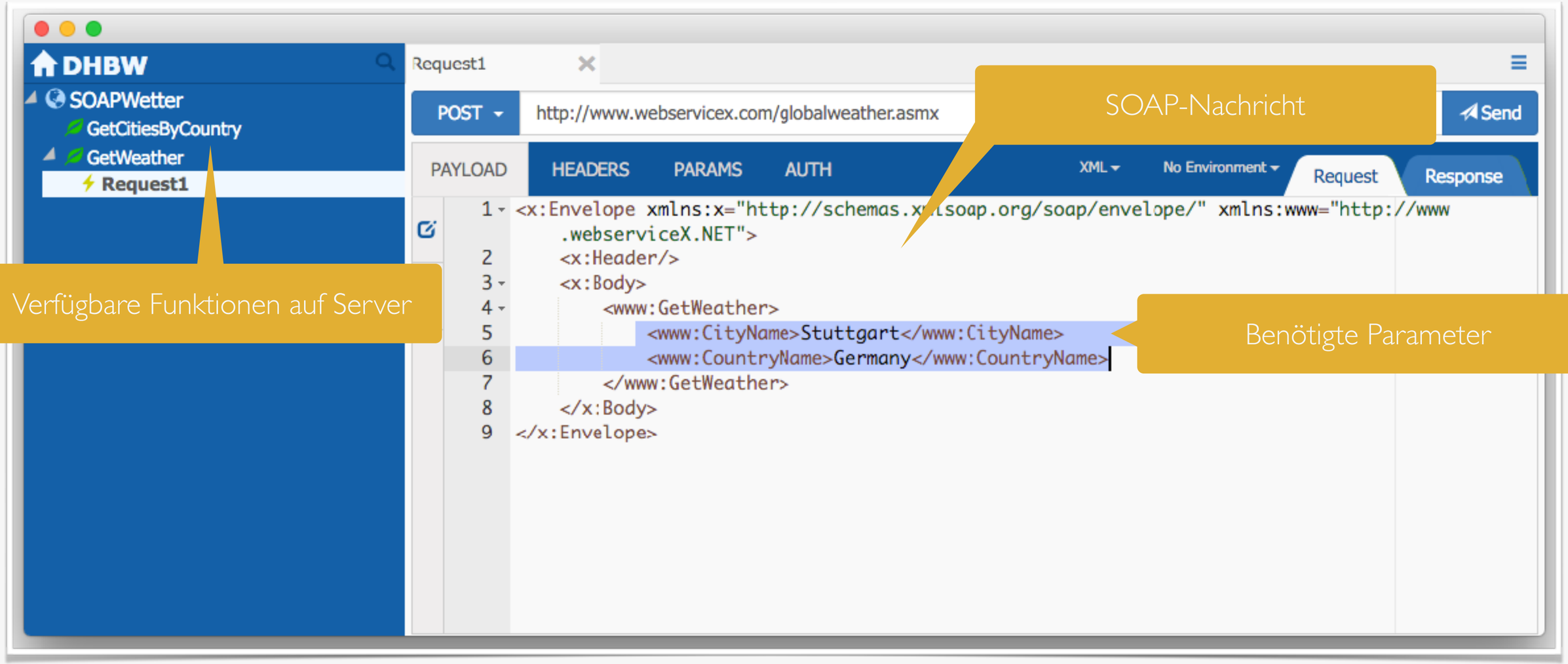
trim/**boomerang**

Boomerang Test-Client



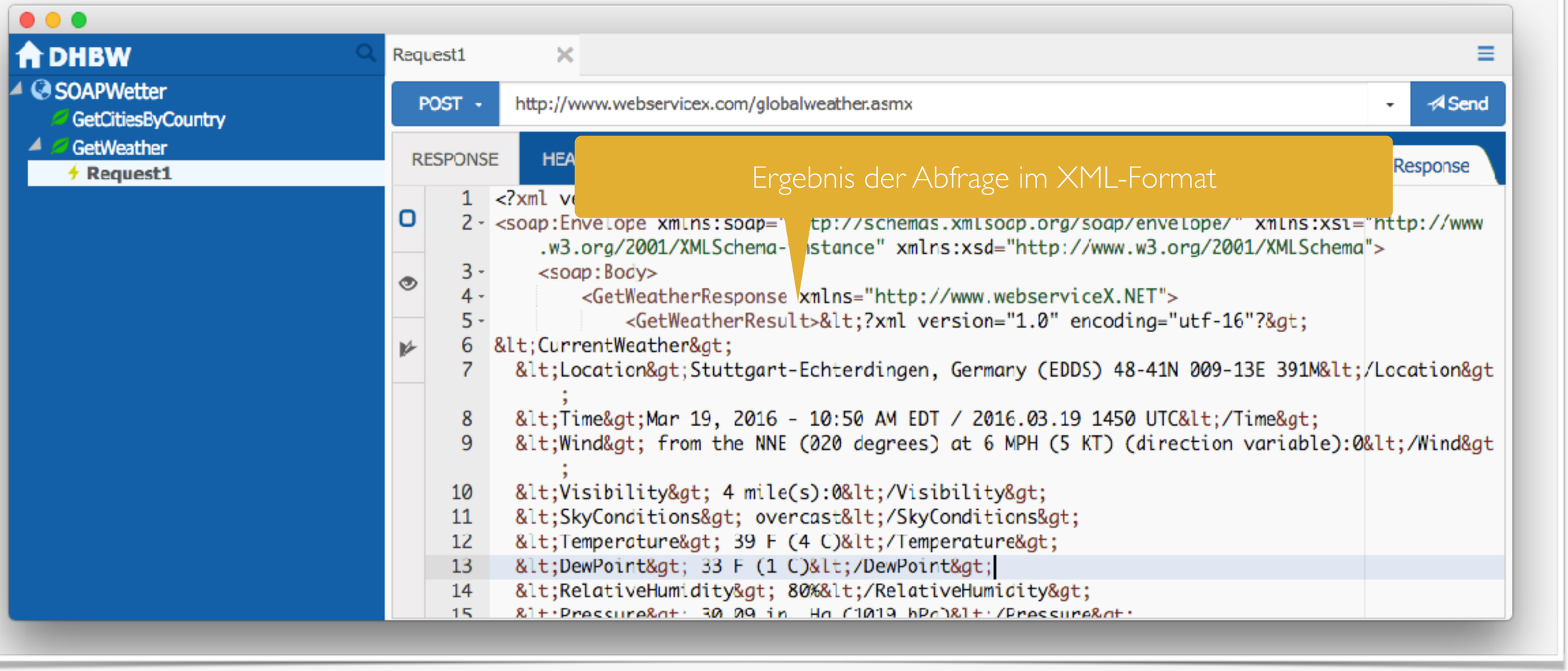
tr.im/amzn

Boomerang Test-Client: Request



tr.im/amzn

Boomerang Test-Client: Response



trim/amzn

REST: Allgemein

- REST steht für Representational State Transfer
- Kein Standard wie SOAP
- Plattform- und programmiersprachenunabhängig
- Auch geeignet für die Erstellung von Web Services
- Bei REST-Anwendungen ist jede Ressource, z.B. ein Artikel in einem Onlineshop, über eine URI adressiert und kann so angesprochen werden
- Transportprotokoll ist HTTP
- Inhalte werden untereinander verlinkt, um dem Client die Möglichkeit zu geben, von einem Zustand in den nächsten zu wechseln

REST: Merkmale

- Ein REST-Netzwerk besteht auch aus einem Client und einem Server
- Ein „REST-System“ besteht aus Ressourcen die per URI adressiert werden
- Jede Anfrage muss alle notwendigen Informationen für die Durchführung beinhalten (HTTP ist zustandslos)
- Es werden die bekannten HTTP-Methoden verwendet (bspw. GET/POST/PUT/DELETE)
- Eine HTTP-Anfrage hat immer den folgenden Aufbau

  Methode ID Version

REST: Beispiel

- Eine Schnittstelle für einen Onlineshop soll entwickelt werden
  - Es soll möglich sein, eine Liste der vorhanden Artikel abzurufen
  - Detaillierte Informationen zu einem Artikel abzurufen
  - Eine Bestellung zu senden

REST: Artikelliste anfordern

- Der REST-Service definiert eine Adresse, unter der die Liste (Ressource) angefordert werden kann
- Der Client kann diese Liste per HTTP-GET abrufen

```
GET /artikel HTTP/1.0
```

```
<?xml version="1.0"?>
<p:Artikeliste xmlns:p="http://www.domain.com" xmlns:xlink="http://www.w3.org/1999/xlink">
    <Artikel id="001" xlink:href="http://www.domain.com/artikel/001"/>
    <Artikel id="002" xlink:href="http://www.domain.com/artikel/002"/>
    <Artikel id="003" xlink:href="http://www.domain.com/artikel/003"/>
    <Artikel id="004" xlink:href="http://www.domain.com/artikel/004"/>
</p:Artikeliste>
```

/dpetrasch/DHBW

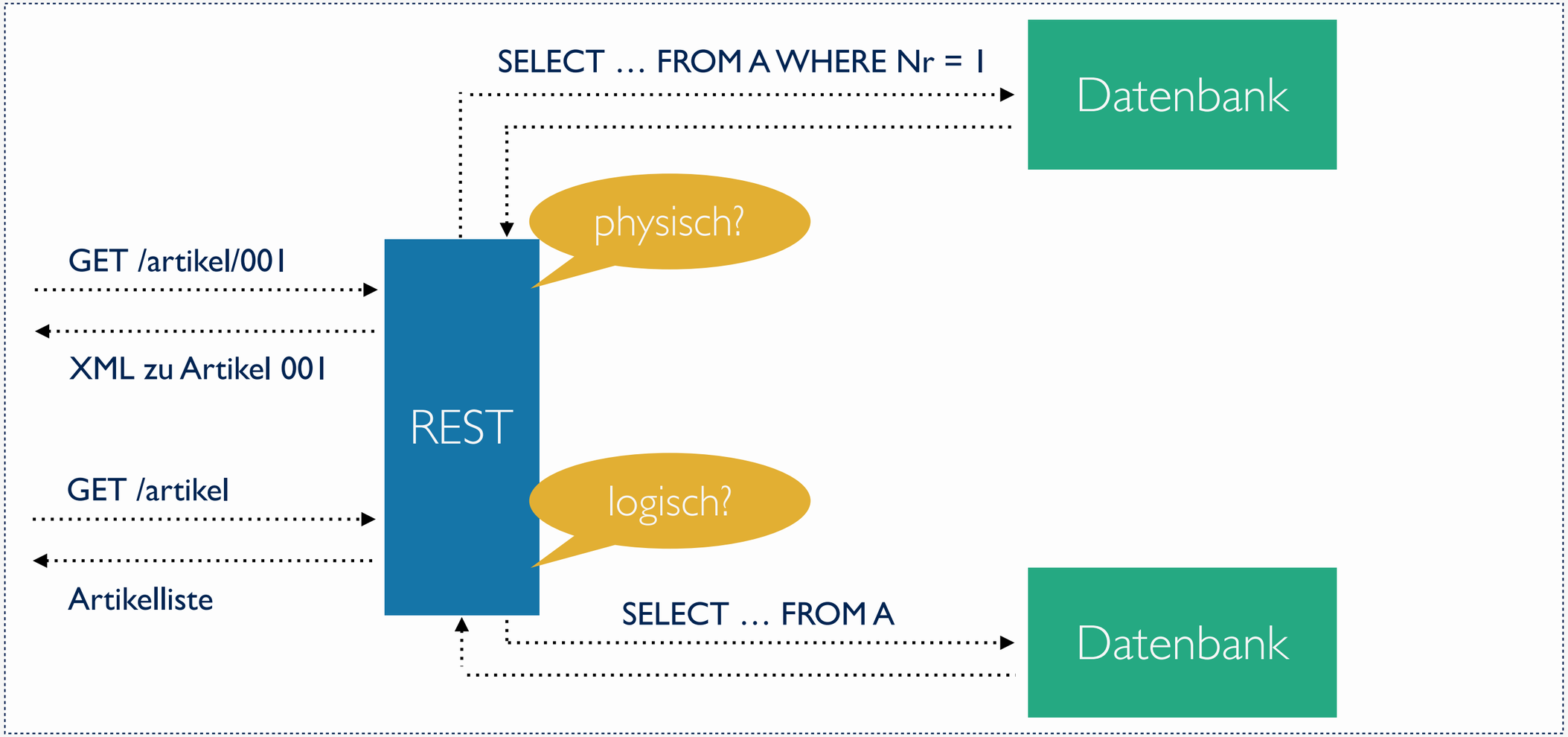
REST: Artikel anfordern

- Der Web Service ordnet jedem Artikel eine eindeutige URL zu, dem Artikel 001 bspw: http://www.domain.com/artikel/001
- Details zu einem Artikel können über ein GET auf diese Url abgerufen werden

```
GET /artikel/001 HTTP/1.0

<?xml version="1.0"?>
<p:Artikel xmlns:p="http://www.domain.com" xmlns:xlink="http://www.w3.org/1999/xlink">
    <Artikelnummer>001</Artikelnummer>
    <Name>Testartikel</Name>
    <Stueckpreis>12.43</Stueckpreis>
    <Menge>115</Menge>
</p:Artikel>
```

/dpetrasch/DHBW

REST: Unterscheidung logische und physische Ressourcen



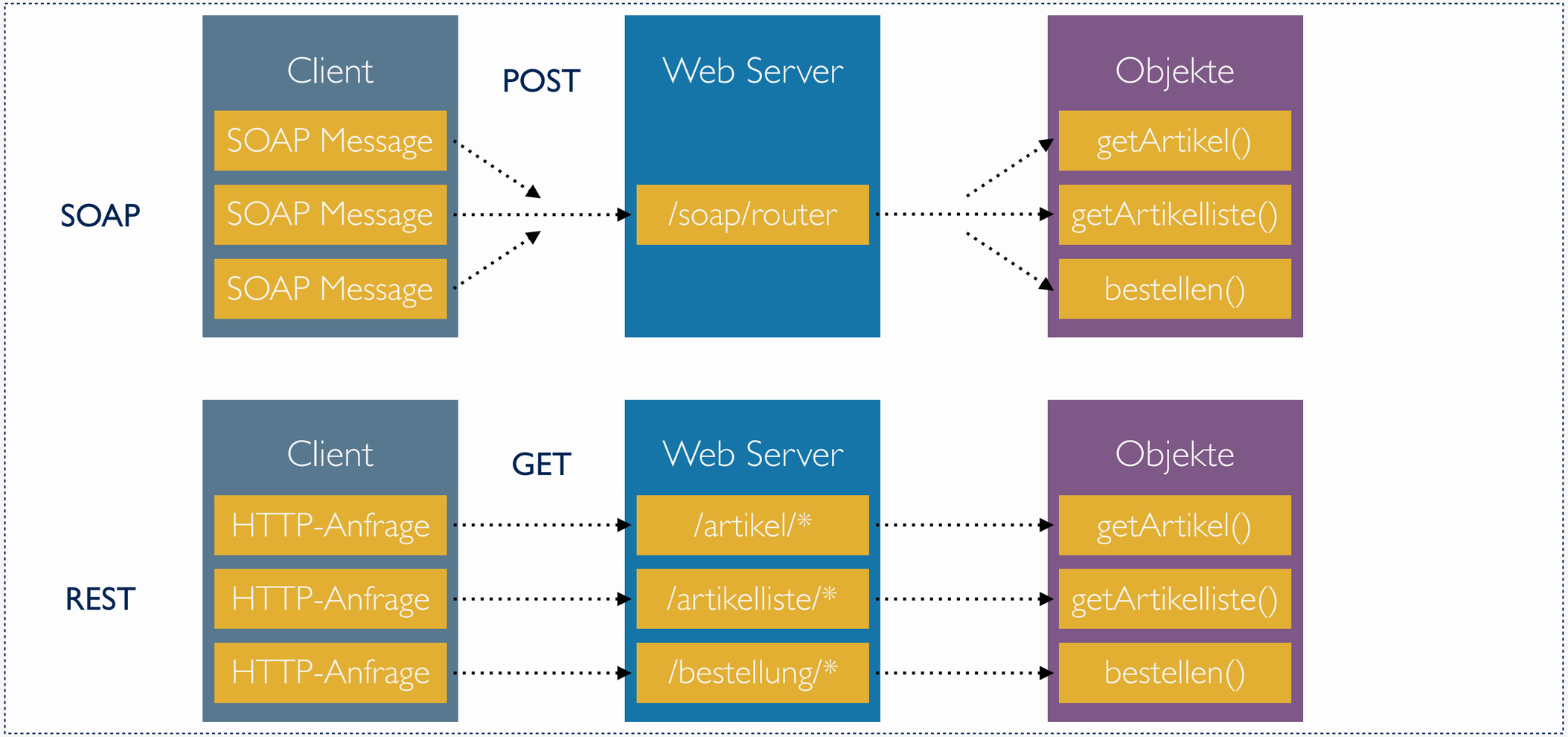
REST: Komplexe Url

- Komplexe Urls können logische Ressourcen filtern
  - Anzahl der zurückzugebenden Elemente begrenzen
  - Filtern nach einem bestimmten Merkmal (bspw. Kategorie oder Preis)
- In dem Beispiel werden alle Artikel mit einer Menge von 10 abgerufen

```
GET /artikel/?menge=10 HTTP/1.0

<?xml version="1.0"?>
<p:Artikel xmlns:p="http://www.domain.com" xmlns:xlink="http://www.w3.org/1999/xlink">
    <Artikelnummer>001</Artikelnummer>
    <Name>Testartikel</Name>
    <Stueckpreis>12.43</Stueckpreis>
    <Menge>10</Menge>
</p:Artikel>
```

/dpetrasch/DHBW

Funktionsweise: SOAP und REST



Web Services: Sicherheit

- Web Services passieren Firewalls problemlos, da HTTP verwendet wird
- Inhalt der SOAP-Nachricht ist innerhalb des Bodys kodiert (als XML) und Administratoren können nicht sehen, was aufgerufen wird
- Bei REST ist die gesamte Nachricht in der URL kodiert. Firewalls können Elemente sperren
- HTTP-Funktionen PUT und DELETE können vom Server gesperrt sein
- Web Services sollten immer über HTTP kommunizieren, da die Daten unverschlüsselt über das Netz übertragen werden

Web Services: Skalierbarkeit und Effizienz

- SOAP-Nachrichten können nicht geloggt werden, da sie über POST verschickt werden
- REST-Anfragen können aufgeteilt werden, dadurch Erhöhung der Performance
- Bei REST sind alle Anfragen zustandslos (stateless)
  (positive Auswirkungen auf Skalierbarkeit)

Einführung in Node.js

Exkurs: Basiswissen Javascript

Variablen

- Variablen speichern Ziffern, Buchstaben, Zahlen, Gleitkommazahlen oder komplexe Dateistrukturen
- Variablen in Javascript sind nicht stark typisiert, eine Unterscheidung von Integer, String, oder ähnlich ist bei der Deklaration nicht notwendig
- Variablen sind ohne Zuweisung (Initialisierung) eines Wertes ‚undefined'

```
// Variablen
var eineVariable = 10;
var einText = "Beispieltext"
var einBoolean = true;
var wertIstUndefined;
```

/dpetrasch/DHBW

Scope-Bereich

- Variablen sind nur in bestimmten Bereichen gültig
- Am Ende eines Bereichs (Scopes) wird der Speicher bereinigt
- Es können auch globale Variablen definiert werden, die überall gültig sind

```
// Code in diesem Bereich kann NICHT
// auf "autoName" zugreifen
function eineFunktion() {
    var autoName = "BMW";
    // Code in diesem Bereich kann auf die
    // Variable "autoName" zugreifen
}
```

/dpetrasch/DHBW

Globale Variablen

- Globale Variablen werden ohne „var“ deklariert
- So implementierte Variablen werden dem Namespace „window“ hinzugefügt

```
// Code in diesem Bereich kann JETZT AUCH
// auf "autoName" zugreifen

function eineFunktion() {
    autoName = „BMW“;  // Kein var!
    // Code in diesem Bereich kann auf die
    // Variable "autoName" zugreifen
}
```

/dpetrasch/DHBW

Rechenoperationen

- Zur Erhöhung oder Verminderung um 1 kann eine verkürzte Schreibweise genutzt werden

```
var ergebnis  = 10 + 4 // ergebnis  = 14
var ergebnis2 = 15 * 2 // ergebnis2 = 30

var zahl = 1;
var eins = ++zahl;  // Pre-Inkrement : eins = 2, zahl = 2
var zwei = zahl++;  // Post-Inkrement: zwei = 2; zahl = 3
```

/dpetrasch/DHBW

Vergleichsoperatoren

- Können zur Überprüfung eines Werts verwendet werden
- Bspw. nach dem Aufruf einer Funktion kann so der Rückgabewert der Funktion überprüft werden

```
// Vergleichsoperationen
var wahr = 1;
var falsch = 0;

console.log( "Ist Wahr gleich   Falsch?     : " + (wahr == falsch) );   // Falsch
console.log( "Ist Wahr ungleich wahr?       : " + (wahr != wahr)   );   // Falsch
console.log( "Ist Wahr groesser als falsch? : " + (wahr > falsch)  );   // Richtig
console.log( "Ist Wahr kleiner  als falsch? : " + (wahr < falsch)  );   // Falsch
```

/dpetrasch/DHBW

Funktionen

- Dienen zur Lösung von Teilproblemen und strukturieren Quellcode
- Machen Quellcode wartungsfreundlicher
- Können ein Objekt zurückgeben
- Beim Aufruf können Parameter übergeben werden

```
// Funktionen
function multiplizieren(num1,num2) {     // num1 und num2 sind Parameter
  var result = num1 * num2;              // Zugriff auf Parameter
  return result;                         // Ergebniswert zurueckgeben
}

// Ausgabe in Konsole, auch Umbruch der Zeile ist möglich
console.log("Ergebnis von 2 * 3 lautet: " +  multiplizieren(2,3));
```

/dpetrasch/DHBW

Kontrollstrukturen

- Kontrollstrukturen ermöglichen die Verzweigung des ansonsten linearen Programm-/Skriptablaufs
- Können den Aufruf von einer Funktion mit unterschiedlichen Parametern vereinfachen
- Können für bestimmte Szenarien kombiniert und verschachtelt werden

Kontrollstrukturen

```
// Kontrollstrukturen
var WertIstWahr   = true, WertIstFalsch = false;

if (WertIstFalsch) {
    console.log(,hello!');  // Dieser Quellcode wird nie erreicht
}

if (WertIstFalsch) {
    // Dieser Scope wird nie erreicht
} else {
    if (WertIstWahr) {
        // Dieser Scope wird erreicht, da der Ausdruck Wahr ist
    } else {
        // Dieser Scope wuerde erreicht werden,
        // wenn WertIstWahr und WertIstFalsch beide "falsch" waeren
    }
}
```

/dpetrasch/DHBW

Kontrollstrukturen

```
var autoMarke = „Porsche";

switch (autoMarke) {
    // Fall: Wert ist gleich "VW"
    case 'VW':
        console.log(ganzerName + " faehrt einen VW");
    break;  // Fallunterscheidung verlassen
    case 'Audi':
        console.log(ganzerName + " faehrt einen Audi");
    break;
    case 'Porsche':
        console.log(ganzerName + " faehrt einen Porsche");
    break;
    default:
        // Falls keine passende Fallunterscheidung vorhanden,
        // kann ein "Catch All"-Fall "default" verwendet werden
        console.log(ganzerName + " faehrt einen " + autoMarke);
    break;
}
```

/dpetrasch/DHBW

Schleifen

- Kopfgesteuerte Schleifen
  - Die Bedingung wird bereits vor dem ersten Durchlauf überprüft. Sollte die Bedingung falsch sein, so wird die Schleife nicht durchlaufen
- Fußgesteuerte Schleifen
  - Die Schleife wird mindestens einmal durchlaufen, da die Bedingung erst am Ende überprüft wird
- Zählschleife (for-Schleife)
  - Die Schleife hat die selben Eigenschaften wie die kopfgesteuerte, hat jedoch die Besonderheit, einen Laufindex anzupassen

For-Schleife

```
// for-Schleife
var index = 0;

// index ist hier die Laufvariable, die vor jedem Durchlauf mit
// der Bedingung index <= 10 verlgeicht wird
// und nach jeder Ausfuehrung um 1 erhoeht wird

for(var index = 1; index <= 10; index = index + 1) {
   console.log("Schleifendurchlauf Nr. " + index);
}
```

/dpetrasch/DHBW

Schleifen

```
// Fussgesteuerte While-Schleife
var i = 0;
do {
  console.log("Schleifendurchlauf Nr. " + i);
  i++;
}
while (i < 10);

// Kopfgesteuerte While-Schleife
i = 0;
while (i < 10) {
   console.log("Schleifendurchlauf Nr. " + i);
  i++;
}
```

/dpetrasch/DHBW

Undefinied und NULL

- Variablen, die nur deklariert und nicht initialisiert werden, haben den Wert „undefined“
- Bei der Zuweisung von Ergebnissen kann eine Variable den NULL-Wert erhalten, wenn die vorherige Funktion einen Fehler verursacht hat
- Grundsätzlich sollte auf „undefined“ und „null“ geprüft werden, wenn unbekannte Drittkomponenten verwendet werden

```
// Undefined
var x;

if (x === undefined) {
    console.log( "x ist undefiniert");
} else {
    console.log( "x ist definiert");
}
```

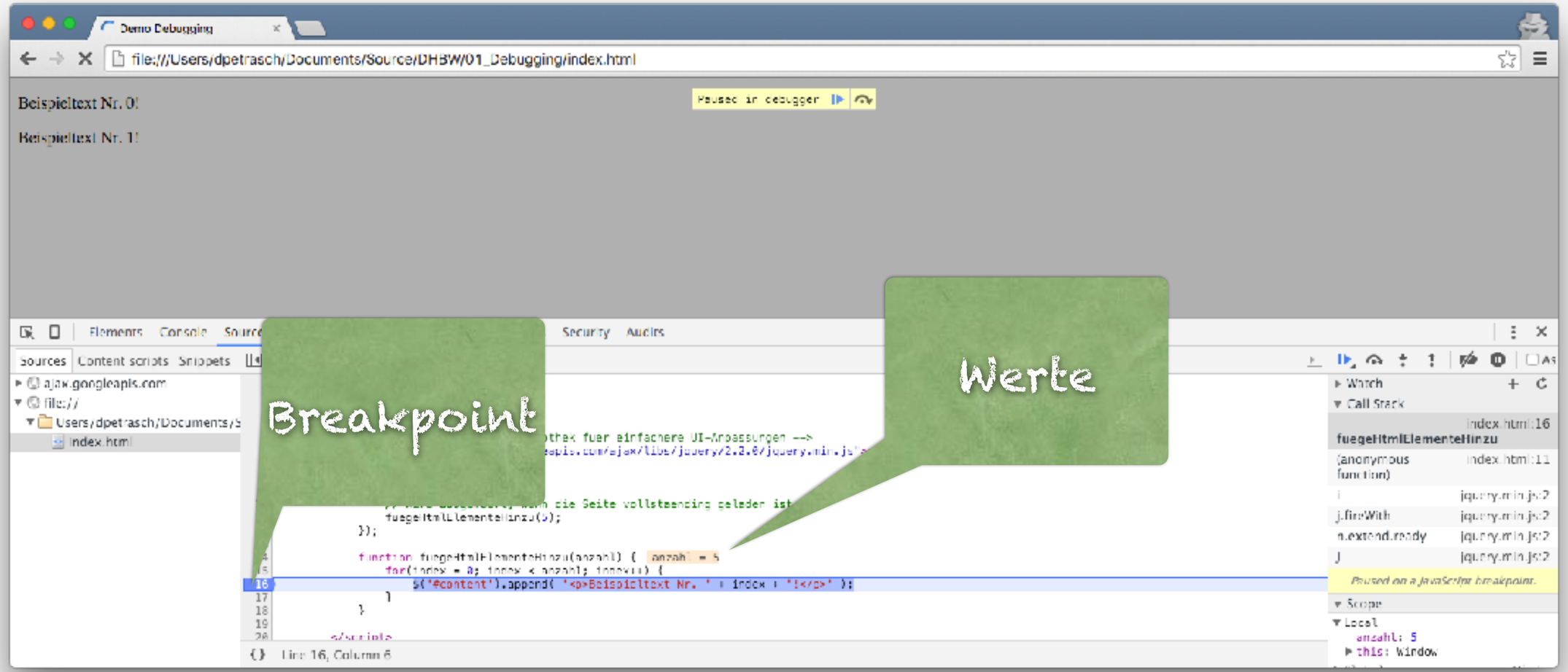
/dpetrasch/DHBW

Debuggen von Javascript-Quellcode

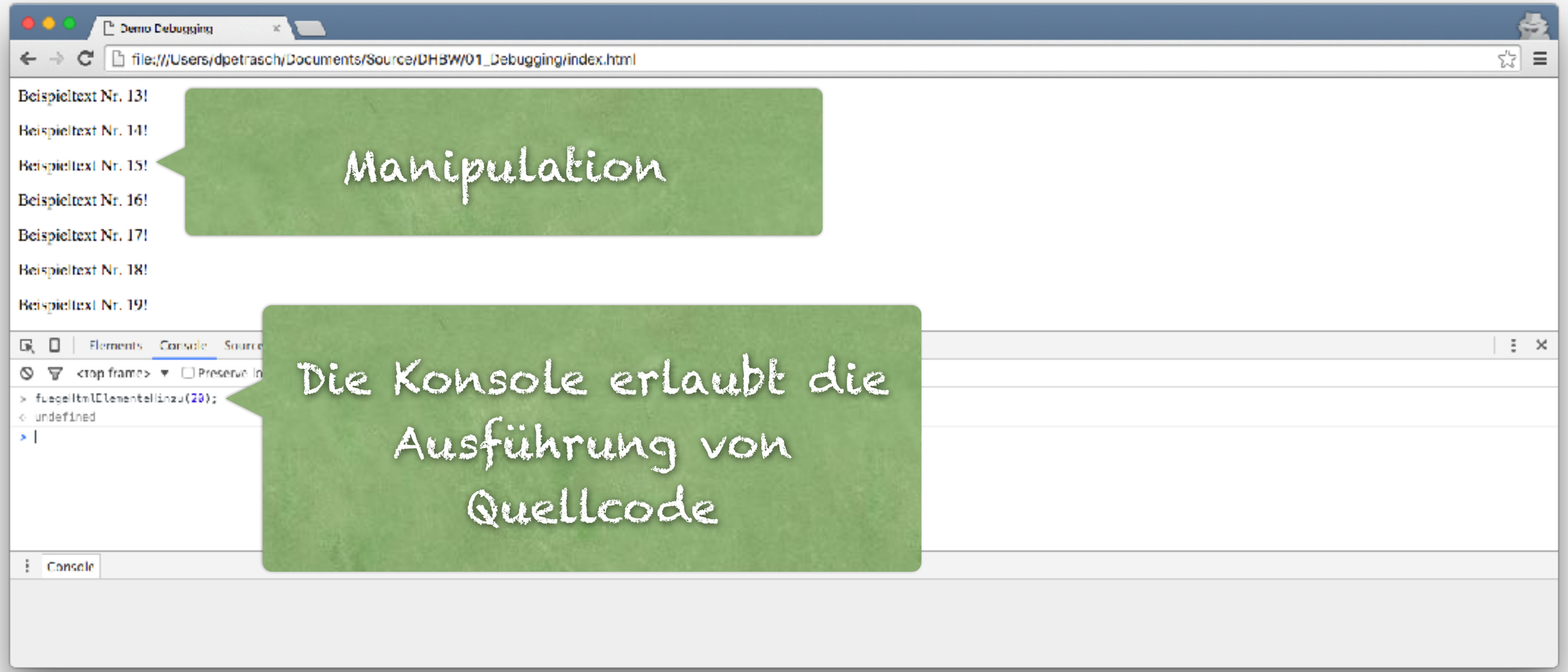
- Javascript-Quellcode, der beim Client ausgeführt wird, wird direkt im Browser ausgeführt und kann „debuggt“ werden - also angehalten und manipuliert werden
- Bei der Entwicklung und das Debuggen wichtig, um Fehler im Quellcode aufzuspüren und beheben zu können
- Die Konsole im Browser erlaubt es Entwicklern, „interaktiv“ mit dem Quellcode Informationen anzuzeigen, auszugeben und zu editieren.

Debuggen von Javascript-Quellcode

Debuggen von Javascript-Quellcode



Manipulieren von Javascript-Quellcode



CoffeeScript und TypeScript

- Zur weiteren Vereinfachung des JavaScript-Quellcodes wurden Sprachen entwickelt, die eine kürzere oder einfachere Schreibweise ermöglicht
- Der Quellcode der Sprache wird anschließend mit einem Compiler zu JavaScript „übersetzt“.
- Die Vereinfachung oder Verkürzung ist oftmals enorm

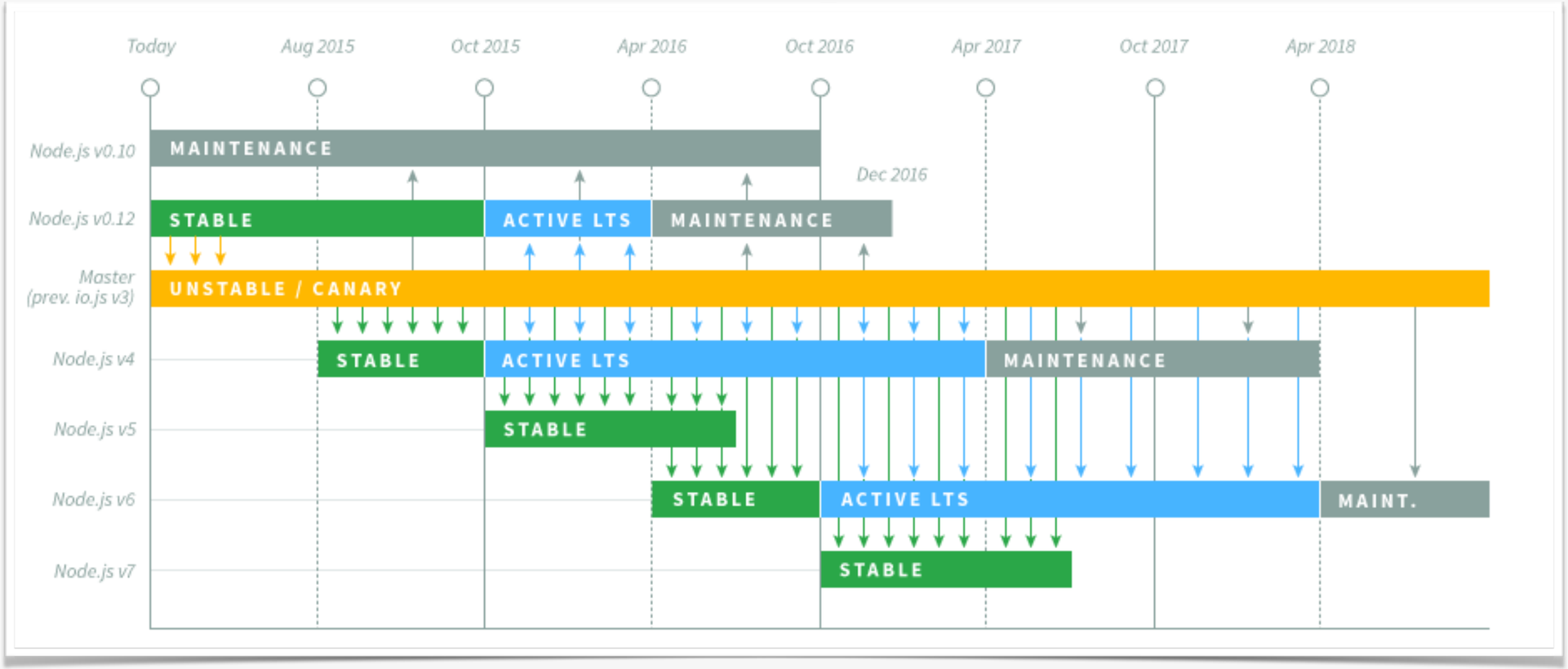
Einführung in Node.js

Allgemeines, Aufbau und Komponenten

Allgemeines

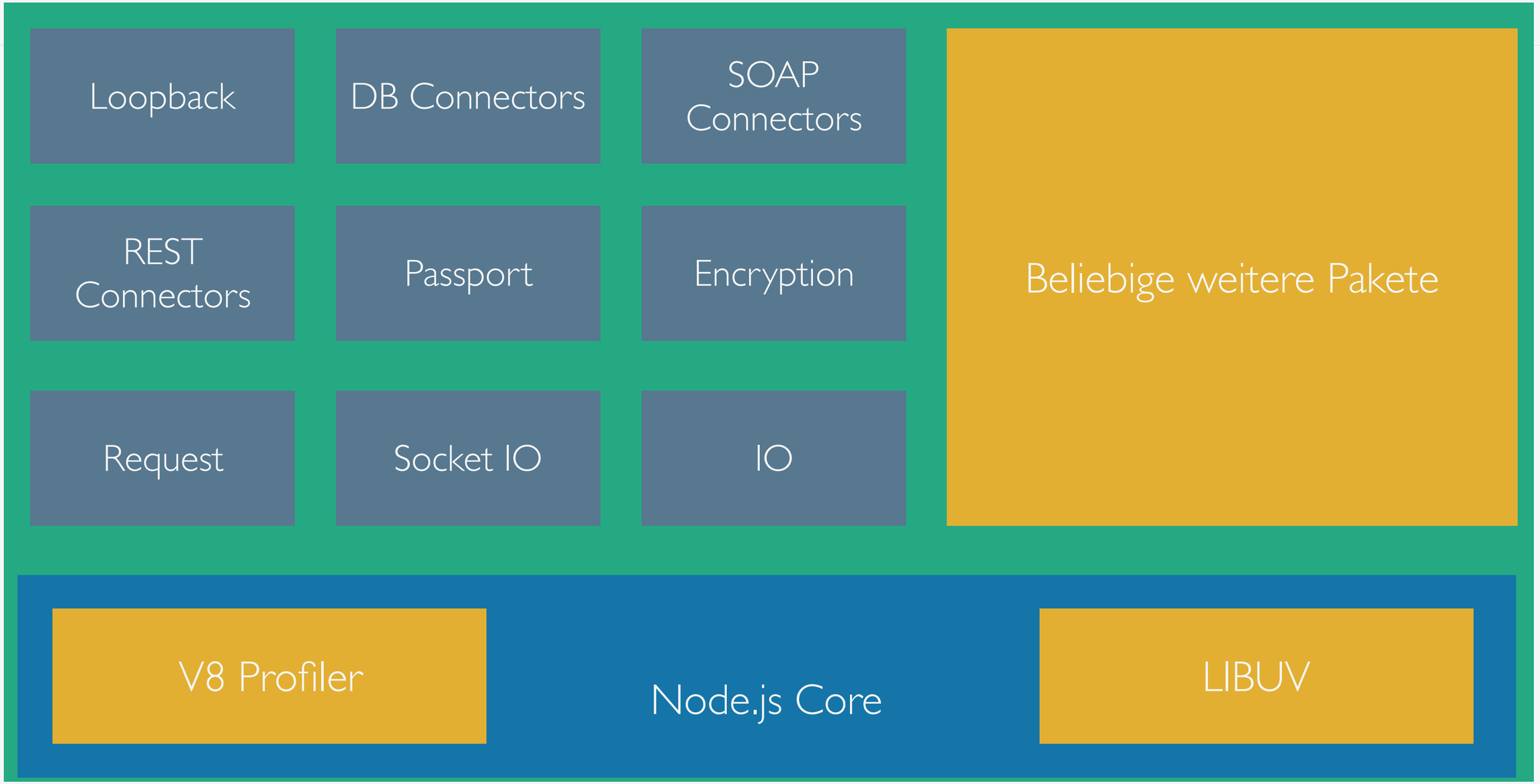
- Node.js ist eine serverseitige Plattform zum Betrieb von Netzwerkanwendungen
- Basiert auf der Google V8-Javascript-Engine
- Erweiterbar ist Node mit Paketen (Modulen)
- Es existiert ein Paketmanager (NPM)
- Das „Grundpaket“ enthält bereits viele APIs und Funktionen
- Es kann auch C++-Quellcode ausgeführt werden

Versionen und Long Term Support



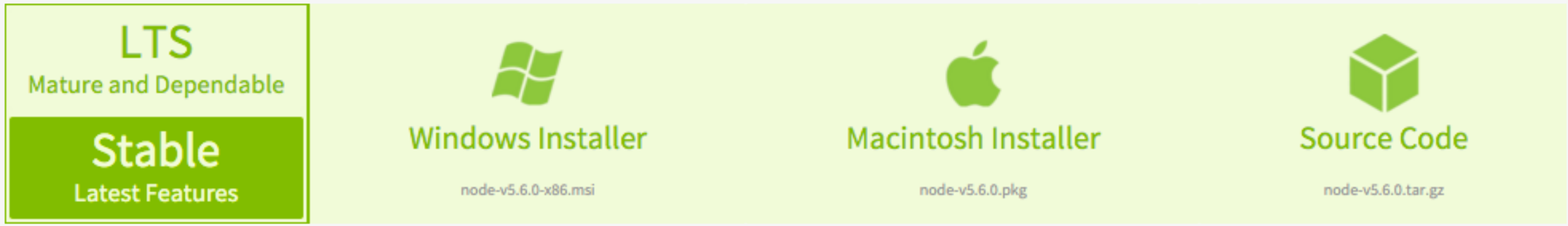
tr.im/nodelts

Aufbau von NodeJS



Node.js Installation

- Für die Entwicklung und Ausführung muss die Node.js Runtime installiert werden
- Diese ist für Windows, Mac, Linux sowie ARM-Geräte verfügbar



Entwicklungsumgebungen

- Nodepad++
- SublimeText
- JetBrains WebStorm (für Studenten kostenlos)

Node.js Verwendung

- Nach der Installation kann „node“ in der Kommandozeile/Terminal aufgerufen werden
- Um Skripte zu starten, werden diese als Parameter aufgerufen

Node.js

Paketmanager

Node Package Manager (NPM)

- Ist vergleichbar mit dem Advanced Packing Tool von Linux (apt)
- Kann Pakete installieren, aktualisieren und entfernen
- Stand Februar 2016 sind über 230.000 Pakete vorhanden

tr.im/npm

package.json

- Liegt in dem Root-Verzeichnis jedes Projekts
- Beinhaltet Informationen zu
  - Verwendeten Paketen
  - dem Projekt
  - Abhänigkeiten
  - möglichen Startbefehlen

tr.im/pjson

package.json

```
{
  "name": „sample-projet",
  "description": "A demo app",
  "version": "1.0.0",
  "private": false,
  "author": „Dennis Alexander Petrasch <dev@dpetras.ch> (http://dpetras.ch)“,
  "repository": {
    "type": "git",
    "url": "https://github.com/dpetrasch/sample.git"
  },
  "engines": {
    "node": ">=5.x"
  },
  "scripts": {
    "start": "NODE_ENV=development ./node_modules/.bin/nodemon server.js",
  },
  "dependencies": {
    "passport": "~0.3.2",
    "passport-facebook": "~2.0.0",
  },
  "devDependencies": {
    "ava": "~0.6.0"
  }
}
```

tr.im/**pjson**

/dpetrasch/DHBW

Lokale und Globale Pakete

- Unterscheiden sich in dem Installationsort
- Lokale Pakete
  - Werden in einem Subordner „node-modules“ am aktuellen Ort installiert
- Globale Pakete
  - Werden in dem Profil des Benutzers im Subordner „node-modules“ installiert
- Unterscheidung anhand

  des Parameters -g

Exkurs: Quellcodeverwaltung

Versionierung und Arbeiten im Team

Gründe für zentrale Quellcodeverwaltung

- Arbeit im Team
- Versionierung
  - Unterschiedliche Stände und Module
- Branch
  - Projekte in Teilprobleme aufteilen und nachher zusammen führen

Systeme

- Subversion
- Git
- Mercurial
- Microsoft Team Foundation Server

Onlineplattformen

- GitHub
  - Studenten erhalten kleinsten Plan umsonst
  - „Crowd Development“
  - Unterschiedliche Projekte
  - „Forken“ und besser machen
- BitBucket
  - Alternative von Atlassen
  - Als „Stash/BitBucket Server“ auch als „on Premise“-Variante erhältlich

Vokabular: Git

- Repository
  - Ein definierter Ordner, zu dem zu bestimmten Zeitpunkten ein Snapshot erstellt wird
  - Snapshot wird durch Benutzer durch einen sog. „Commit“ erstellt
  - Es handelt sich um einen differentiellen Snapshot zur vorherigen Version
- Stage
  - Ein Bereich, in dem Änderungen für Dateien vorgemerkt werden
  - Ein Sammelplatz für Änderungen die dann in ein „Commit“ überführt werden

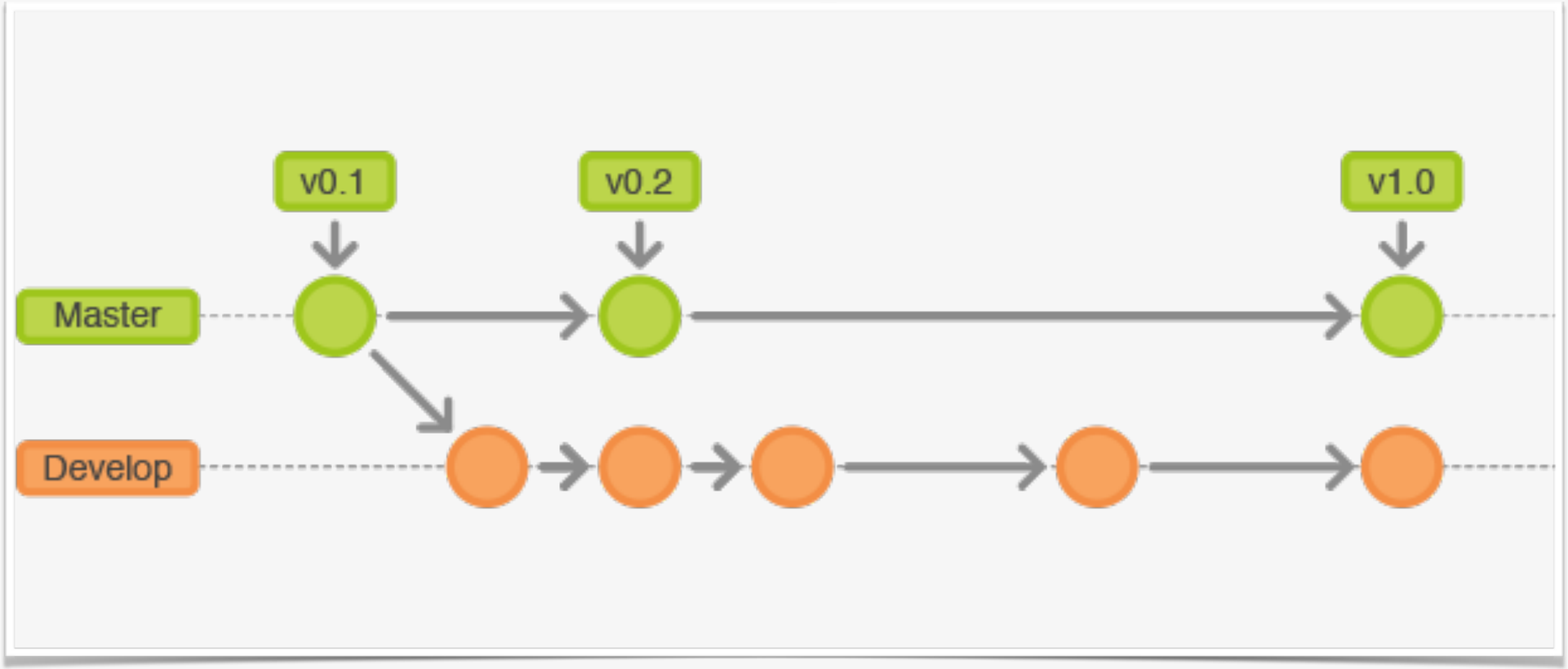
Vokabular: Git

- Commit
  - Alle Änderungen innerhalb der Stage werden mit einer frei definierbaren Beschreibung in den Zeitstahl hinzugefügt
  - Es wird ein neuer „Snapshot“ erstellt
- Branch
  - auch Ast genant
  - Es gibt einen „Urast“, den „master“
  - Ein Ast kann in einen anderen Ast verzweigen

Vokabular: Git

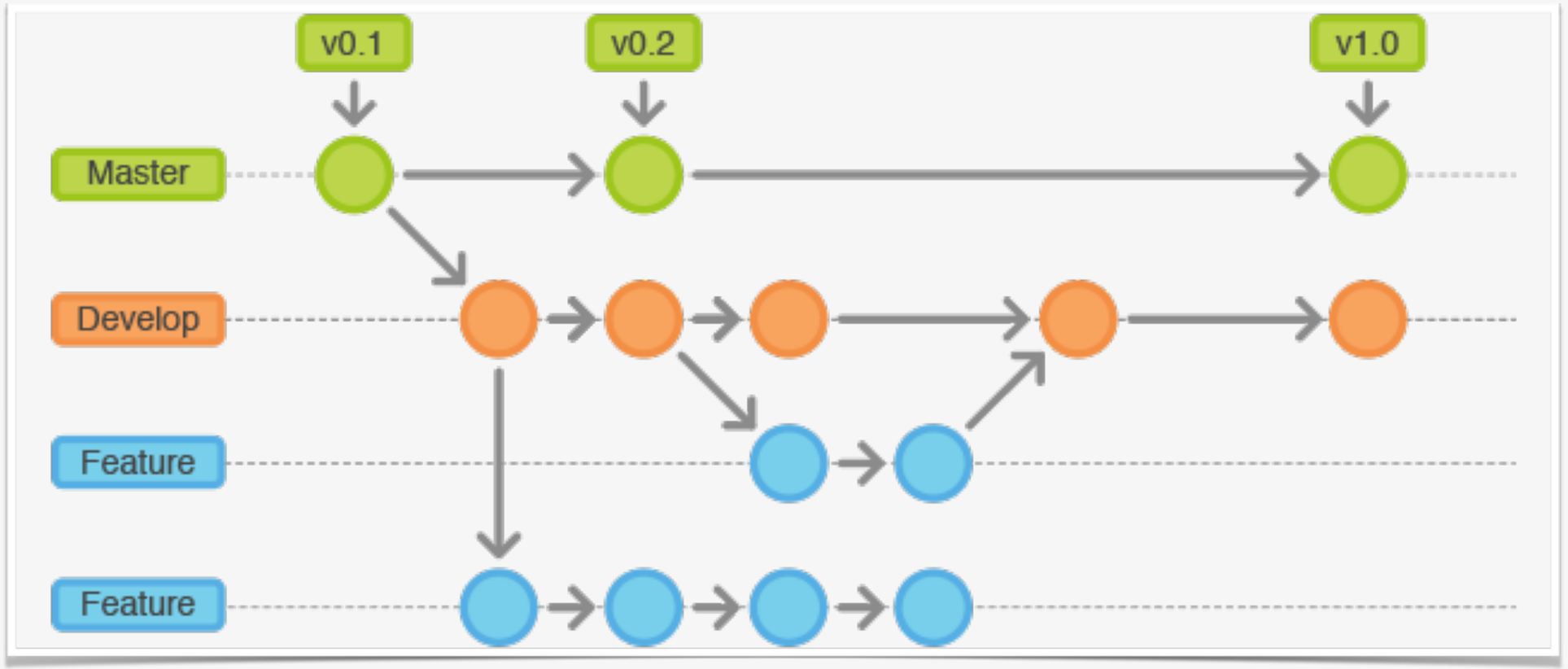
- Revert
  - Es kann zu einem bestimmten Commit in der Zeitlinie gesprungen werden. Die Änderungen an den Dateien wird dann bis zu dem Zeitpunkt rückgängig gemacht
- Init
  - Der Prozess zum Erstellen eines neuen Repositories wird „init“ genannt
- Checkout
  - Ein vorhandenes Repository auf dem System duplizieren
  - Git-Repositories sind überall vollkommen„gespiegelt“! Es gibt keine Server, nur gleichberechtigte Repositories!

Git Workflow: Master und Entwicklung



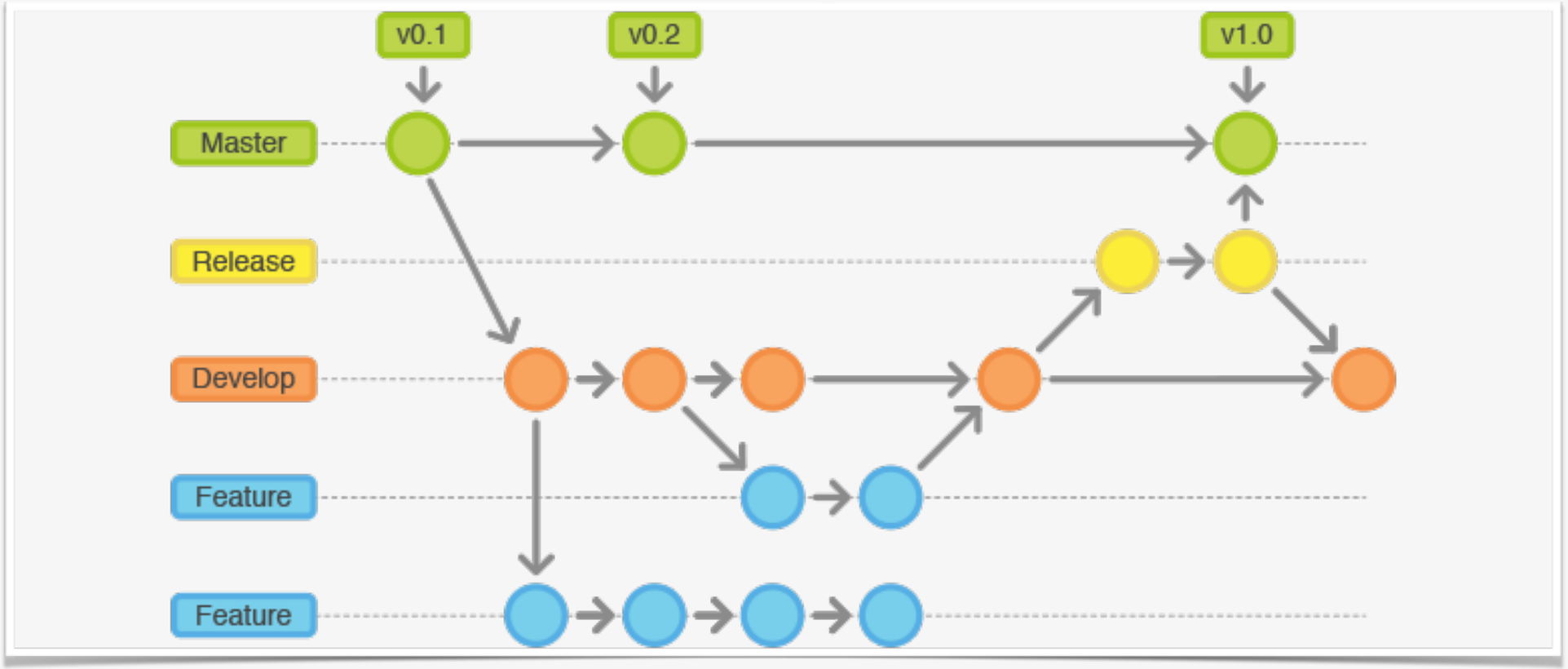
tr.im/git

Git Workflow: Funktionsbranches



tr.im/git

Git Workflow: Funktionsbranchen mit Release



tr.im/git

Grunt

Prozesse während der Entwicklung optimieren

Wiederholende Aufgaben während der Entwicklung

- Quellcode überprüfen (lint)
- Unit Tests ausführen
  - Ergebnisse / Bedingungen auswerten
- Quellcode verkleinern, bzw. „uglifien“
- Kompilieren
- In Test-Umgebung hochladen (Staging Area)
- Test-Datenbanken laden
- Dateiverzeichnis vorbereiten

Hilfsprogramm Grunt

- Plattform für Ausführung von Befehlen
- Konfiguration über Konfig-Datei „Gruntfile“
  - Aufgeteilt in einzelne Aufgabenbereiche
- Aufruf über Parameter, bspw. „grunt test“
- Führt andere Node.JS Komponenten aus und kann Ergebnisse interpretieren

Grunt Konfiguration

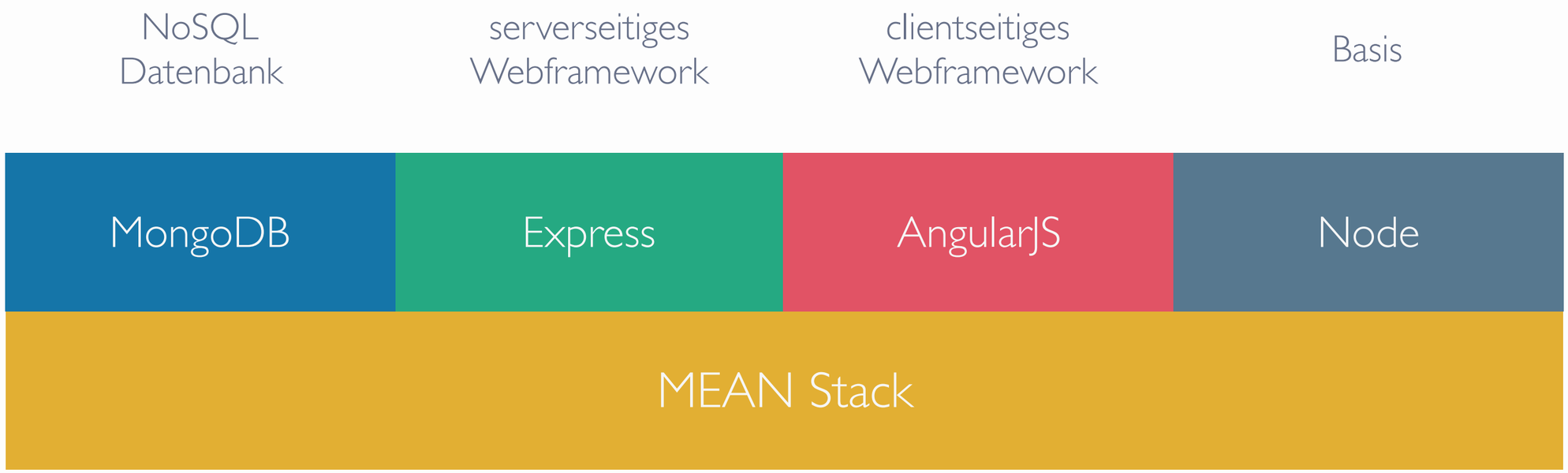
```
module.exports = function(grunt) {

  grunt.initConfig({
    pkg: grunt.file.readJSON('package.json'),

   uglify: {
      options: {
        banner: '/*! <%= pkg.name %> <%= grunt.template.today("dd-mm-yyyy") %> */\n'
      },
      dist: {
        files: {
          'dist/<%= pkg.name %>.min.js': ['<%= concat.dist.dest %>']
        }
      }
    },
    qunit: {
      files: ['test/**/*.html']
    }
  });

  grunt.loadNpmTasks('grunt-contrib-uglify');
  grunt.loadNpmTasks('grunt-contrib-qunit');

  grunt.registerTask('test', ['qunit']);
  grunt.registerTask('default', [qunit', ,uglify']);

};
```

/dpetrasch/DHBW

yeoman - Templates für schnellere Webentwicklung

- Scaffolding
  - Grundlegende Einrichtungsschritte sollen automatisiert werden
    - MVC-Framework
    - Basiscontroller / Views
    - Authentifizierung
    - Inhaltsbereiche (Registrierungsdialog, Login, Mitgliederbereich, o.ä.)
  - Verschiedene Frameworks oder Pakete werden miteinander kombiniert
  - „Beliebte Kombinationen“ können durch Dritte weiterverwendet werden

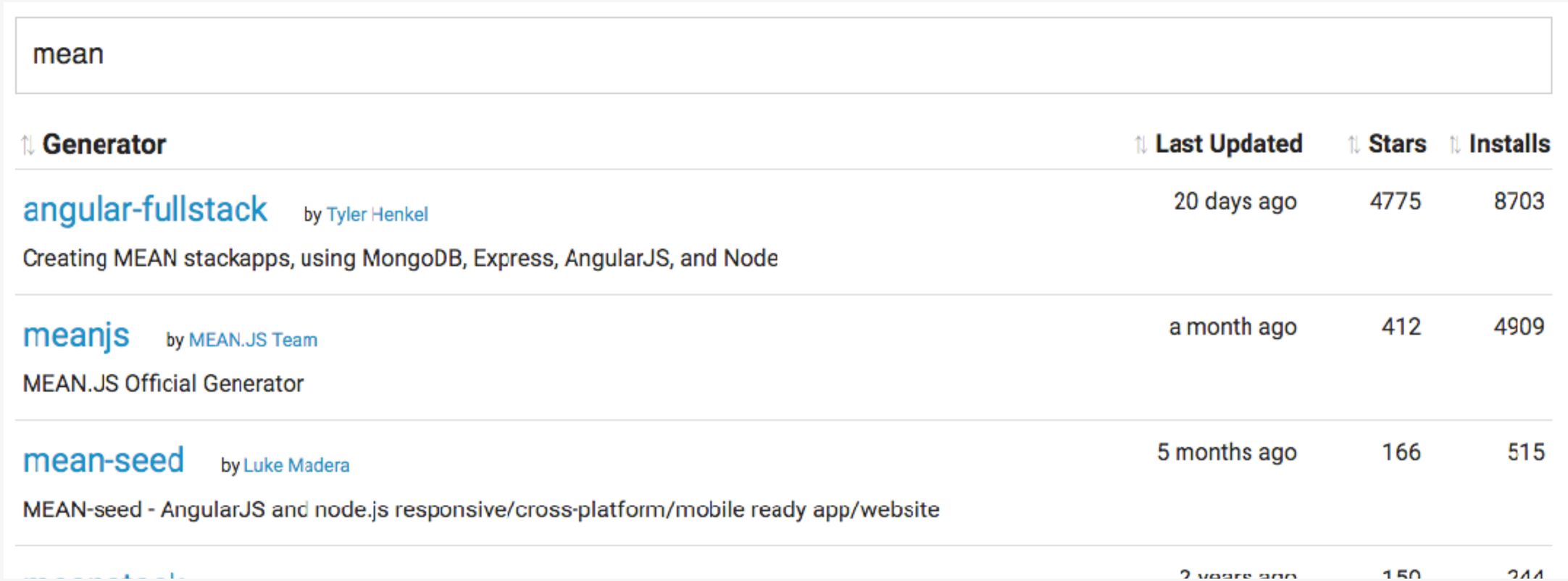
Pakete



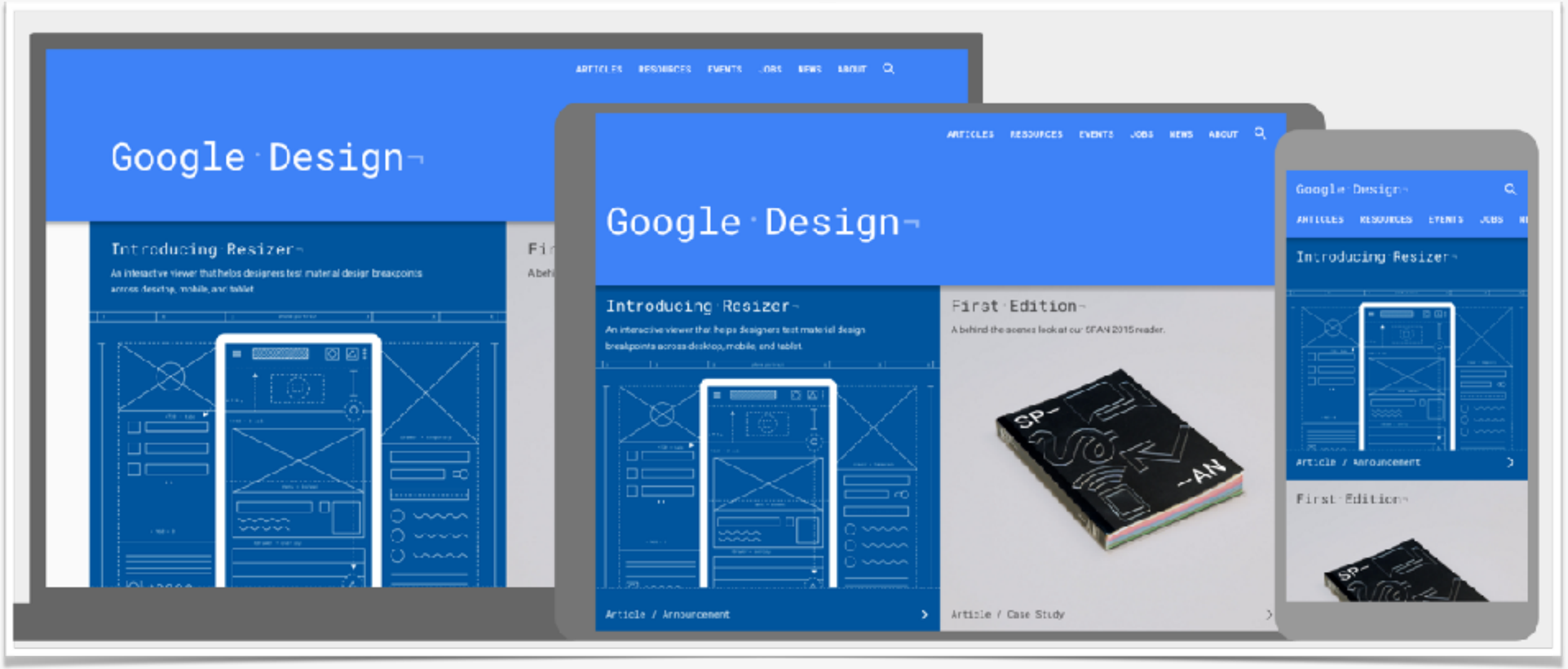
/dpetrasch/DHBW

yeoman - Scaffolding Verzeichnis

- yeoman stellt ein Katalog sowie eine Software zur Konfiguration bereit
  - Webkatalog wird von Mitgliedern gepflegt



Wie responsible ist meine Seite? Google Design: Resizer



trim/resizer