Web Engineering II

serverseitige Webentwicklung

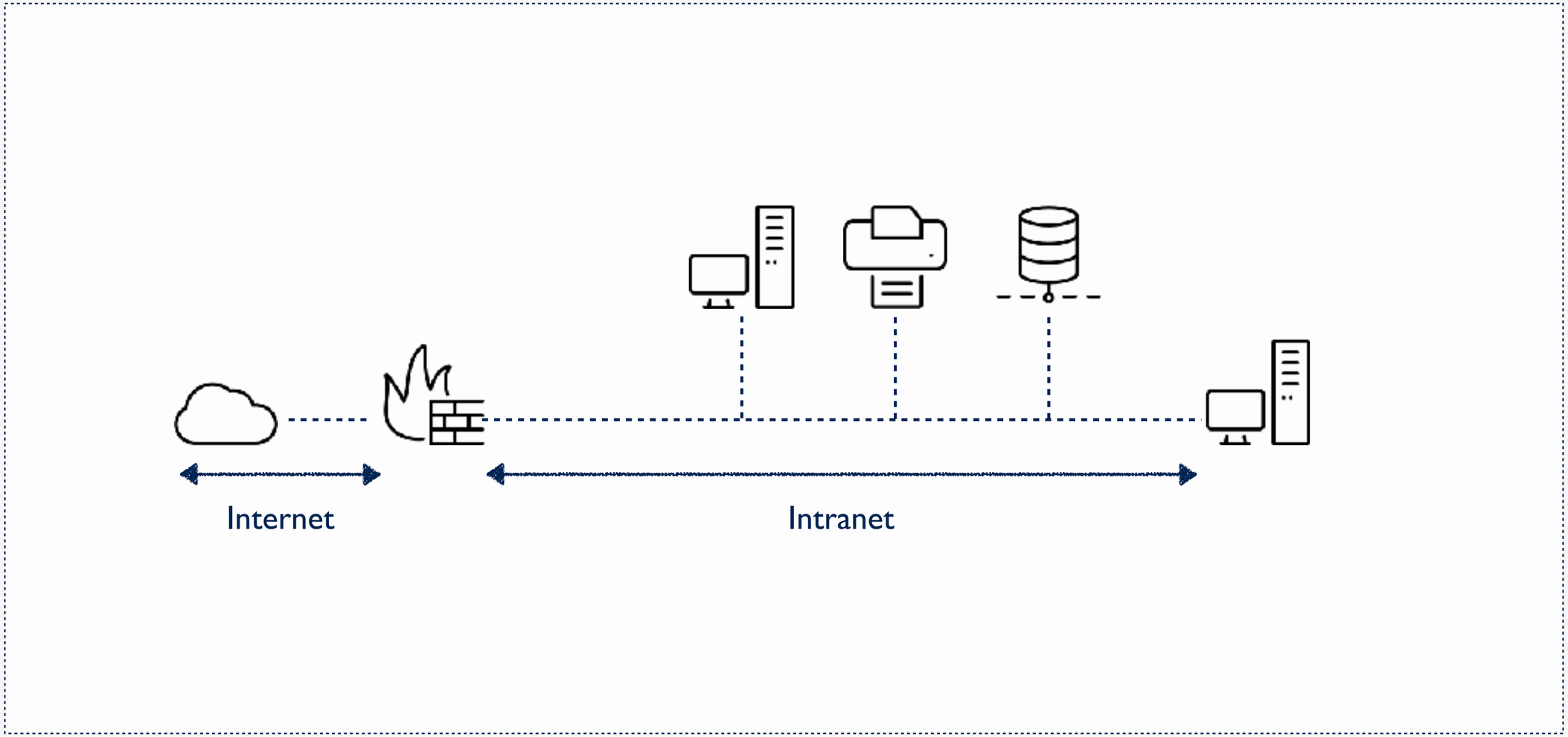
Internet und Intranet

IPv4 und IPv6

Internet

- Webtechnologien bauen auf dem Internet auf.
- Das WWW ist nur ein Teil des Internets
- Technologische Basis des Internets ist das Internet Protocol
- Jeder (direkt im Internet erreichbare) Rechner ist eindeutig durch eine IP-Adresse identifiziert

Internet und Intranet



IPv4 und IPv6

- 32 Bit Adresslänge
- 4 Gruppen zu je 8 Bit
- Darstellung Dezimal
- $4{,}3 \cdot 10^9$ Adressen

00101110.00000100.00011010.01010111

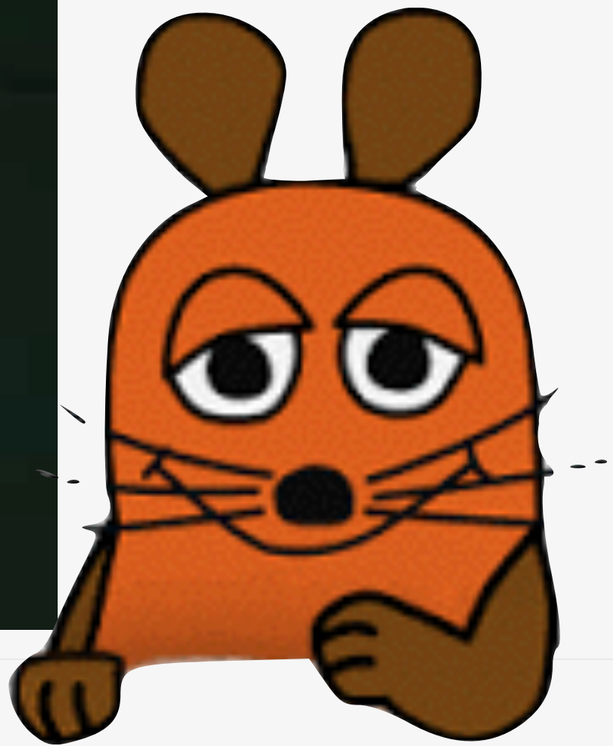
46.4.26.87

- 128 Bit Adresslänge
- 8 Gruppen zu 16 Bit
- Darstellung **Hexa**dezimal
- $3{,}4 \times 10^{38}$ Adressen

2001:4860:A002:0000:0000:0000:0000:0068

2001:4860:A002::68

Wie funktioniert das Internet?



Dual-Stack

IPv4 und IPv6 Anschlüsse

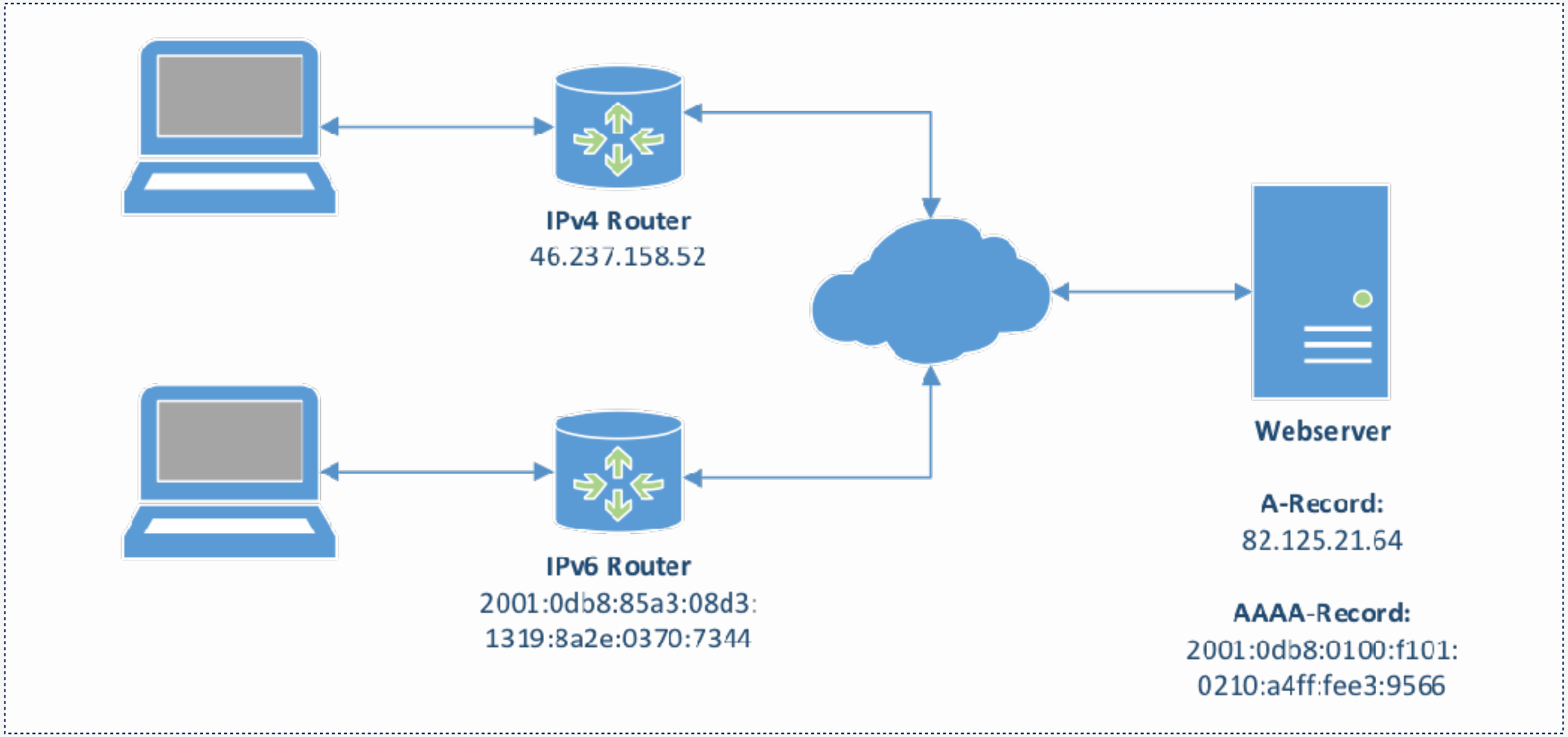
Dual Stack

- Dual-Stack beschreibt den Parallelbetrieb von IPv4 und IPv6
- Es wird eine IPv4 mit vollwertigem NAT verwendet
- Sowie ein IPv6 Netz bereitgestellt

Dual Stack

- Entworfen für den Übergang bis nur noch IPv6 genutzt wird
- Unterstützung aller Geräte notwendig
  - Betriebssysteme
  - Router / Switche
  - Server

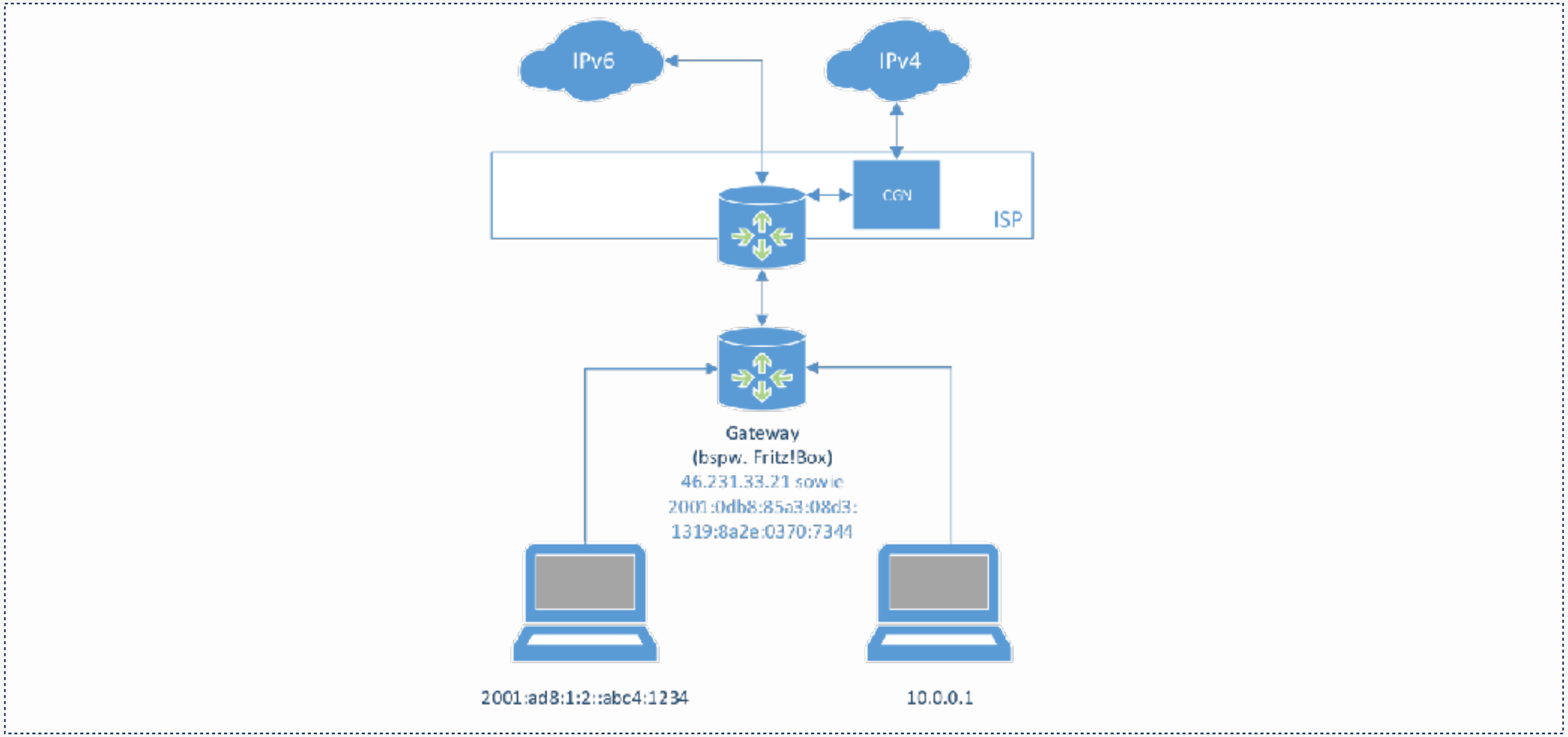
Dual Stack


IPv4 Router
46.237.158.52
IPv6 Router
2001:0db8:85a3:08d3:
1319:8a2e:0370:7344
Webserver
A-Record:
82.125.21.64
AAAA-Record:
2001:0db8:0100:f101:
0210:a4ff:fee3:9566

Dual Stack Lite

- Einschränkung: Carrier Grade NAT
  - Mehrere Nutzer teilen sich eine IPv4-Adresse
  - Kein NAT möglich
  - bspw. Unitymedia-Anschlüsse

Dual Stack Lite



Vorteile

- Nutzern werden auf IPv6 umgestellt
- kompatible IPv6-Dienste können genutzt werden
- zusätzlich Fallback auf IPv4
- Jedes Gerät erhält eine eigene IP-Adresse

Nachteile

- Dual Stack Lite
  - durch Carrier Grade NAT kein eigenes natten möglich
- Webhoster und Cloud-Plattformen unterstützen noch kein IPv6 (AWS, Azure, o.ä.)

Tunnelprovider

- Tunnelprovider ermöglichen IPv4 -> IPv6 Tunnel (Portmapper)
- Somit direkter Zugriff auf IPv6 Geräte über CGN möglich

Tunnelprovider

tr.im/festeip

Client / Server

Dienstbereitstellung

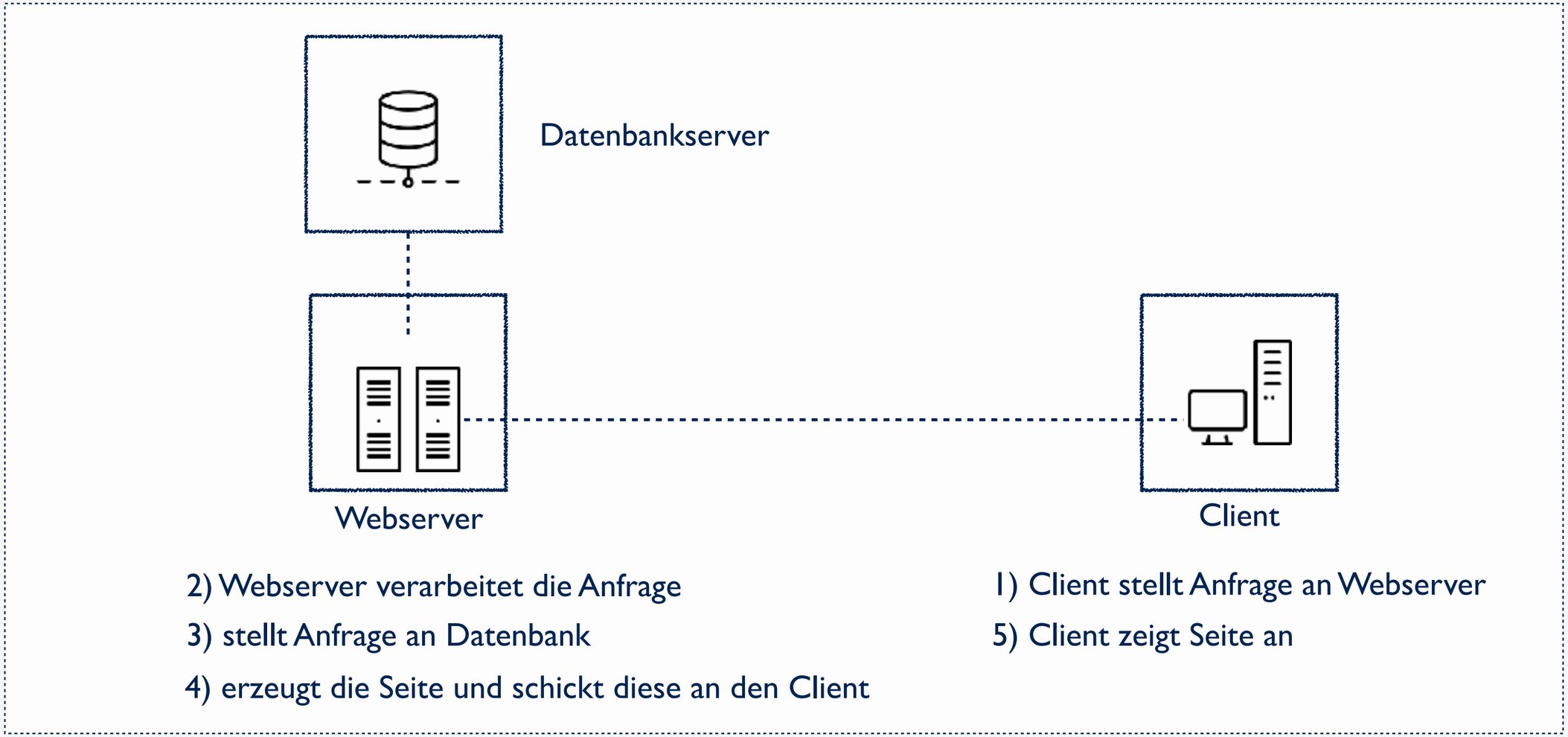
Client / Server

- Anfragen werden von einem Dienstnutzer (Client) an einen Anbieter (Server) gestellt
- Beim Zugriff aufs „Internet“ ist der Client in der Regel ein Endgerät wie z. B. Computer, Smartphones, Tablets, Infotainmentsysteme in Haushalt und Kfz
- Der Anbieter stellt unterschiedliche Dienste für einen oder mehrere Clients bereit
- Backendserver (bspw. Datenbankserver) kommunizieren nicht direkt mit dem Client

Typische Dienste

- File-Server
  - Bereitstellung von Daten über verschiedene Protokolle, in der Regel im LAN
- Datenbankserver
  - Stellt Informationen einer oder mehrerer Datenbanken bereit
  - Wird in der Regel über Anwendungen oder Webdienste angesprochen
- Groupwareserver
  - Stell Zugriff auf Mailpostfächer, Kalender, Kontakte, Workflows etc. bereit
- Mailserver
  - Nimmt E-Mails entgegen und sendet sie weiter, verarbeitet Mails an Hand von Regeln
- Webserver
  - Stellt Informationen über HTTP zur Verfügung, greift dabei auf Daten anderer Server zu

Kommunikation im Web



NAT

Network Address Translation

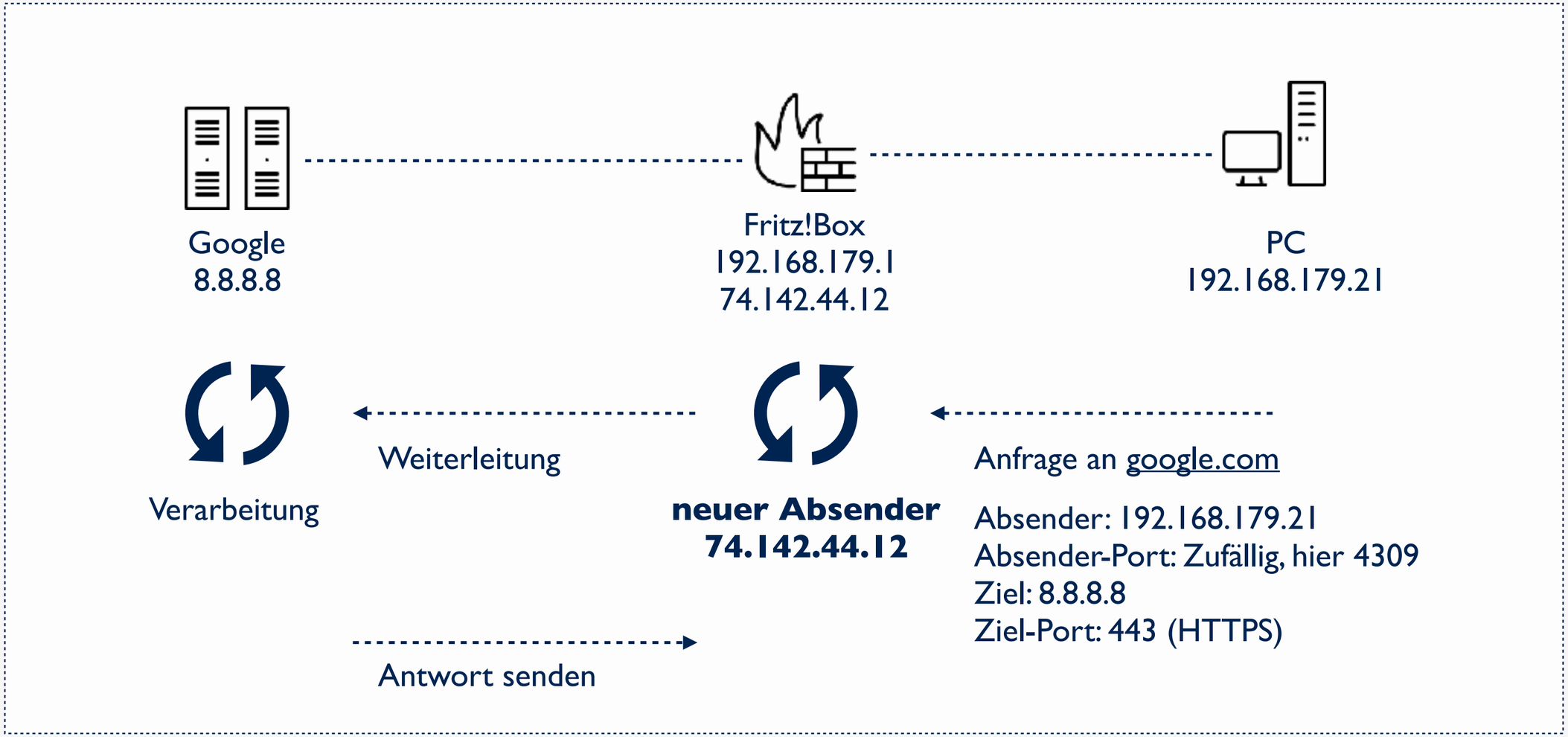
Network Address Translation

- Network Address Translation dient zum Verbinden von unterschiedlichen Netzen
- wird in Source-NAT und Destination-NAT unterteilt

Source-NAT (Masquerading)

- Mehrere Netzwerkgeräte kommunizieren über eine gemeinsame IP-Adresse nach außen. Hierbei wird die Quelladresse geändert.
- wird beispielsweise bei dem heimischen Router verwendet, um mit mehreren Geräte über eine IP-Adresse zu kommunizieren
- Der Router „übersetzt“ und ersetzt die interne Adresse (bspw. 192.168.179.12) mit der gemeinsamen Adresse (74.142.44.12) und ändert die in den Datenpaketen stehenden Quell- und Zieladressen, damit das Paket das Ziel erreicht

Source-NAT

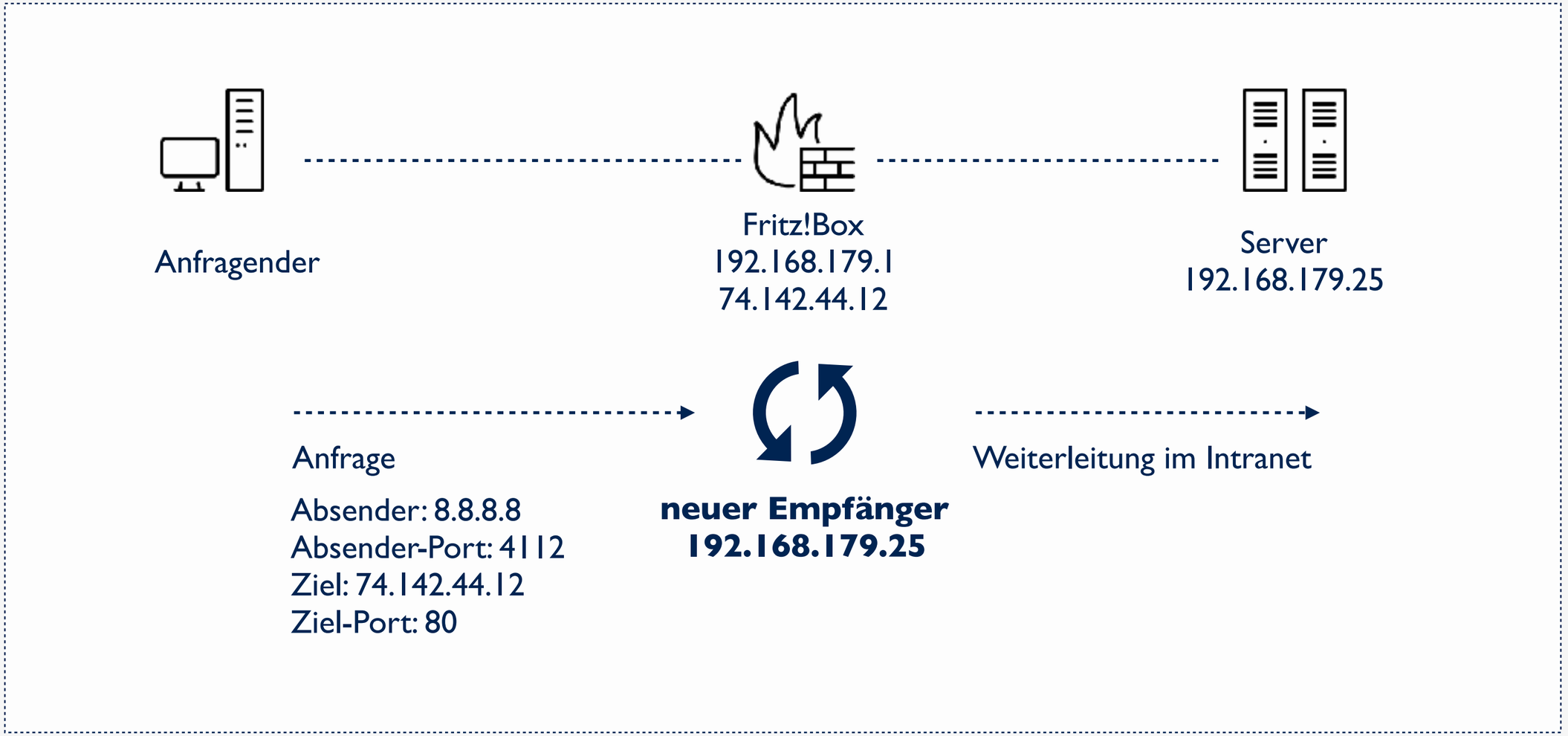

Google
8.8.8.8
Fritz!Box
192.168.179.1
74.142.44.12
PC
192.168.179.21
Verarbeitung
Weiterleitung
neuer Absender
74.142.44.12
Anfrage an google.com
Absender: 192.168.179.21
Absender-Port: Zufällig, hier 4309
Ziel: 8.8.8.8
Ziel-Port: 443 (HTTPS)
Antwort senden

Destination-NAT

- Wird genutzt, um die Zieladresse in eingehenden Paketen zu ändern.

  Pakete an 74.142.44.12:80 werden so beispielsweise zu 192.168.179.25:80 übersetzt
- nötig, um aus dem Internet auf Dienste zuzugreifen, die auf Maschinen mit privaten Adressen arbeiten („Portfreigabe“)

Destination-NAT



Webserver

Übersicht und Marktanteile

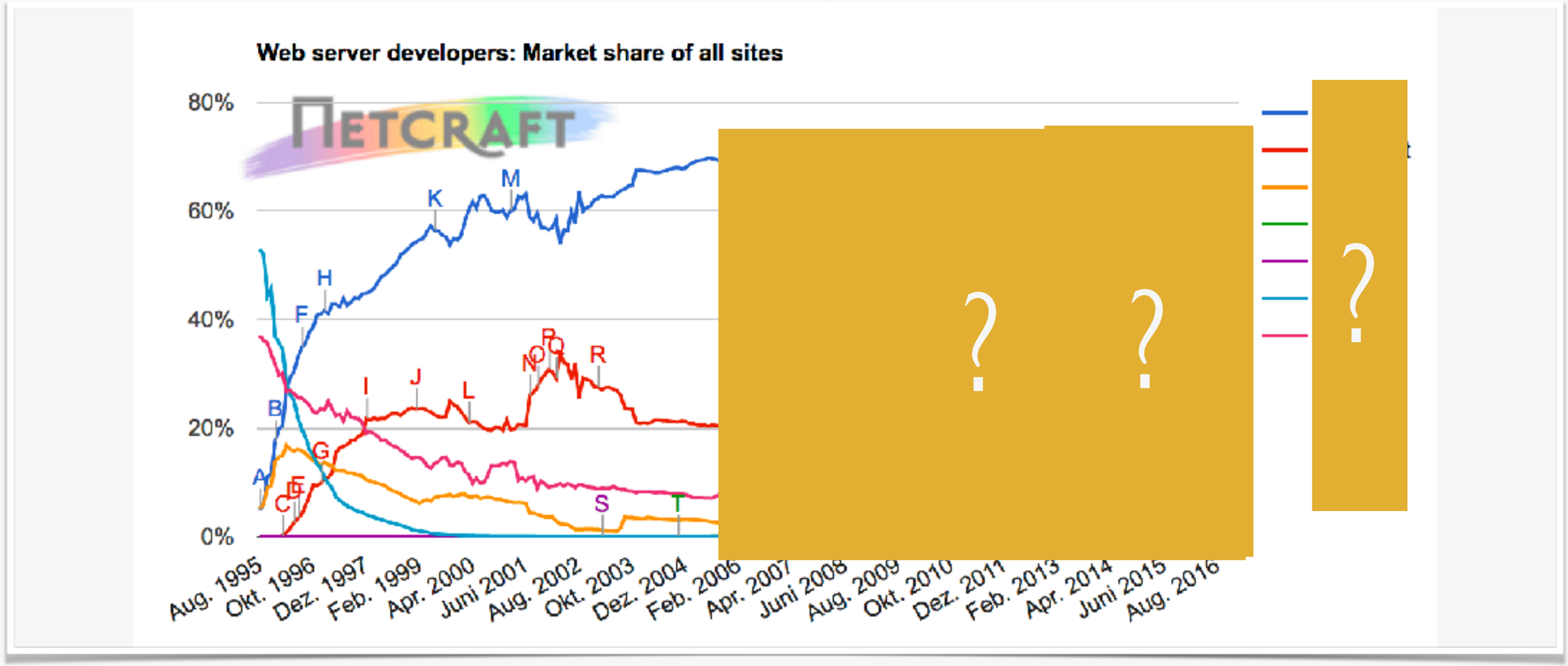
Webserver

- Ein Serverdienst, der Informationen über das HTTP-Protokoll zur Verfügung stellt
- Stellt Informationen über das HTTP-Protokoll zur Verfügung
- TCP Port 80 für HTTP
- TCP Port 443 für HTTPS (SSL/TLS)

Bekannte Webserver

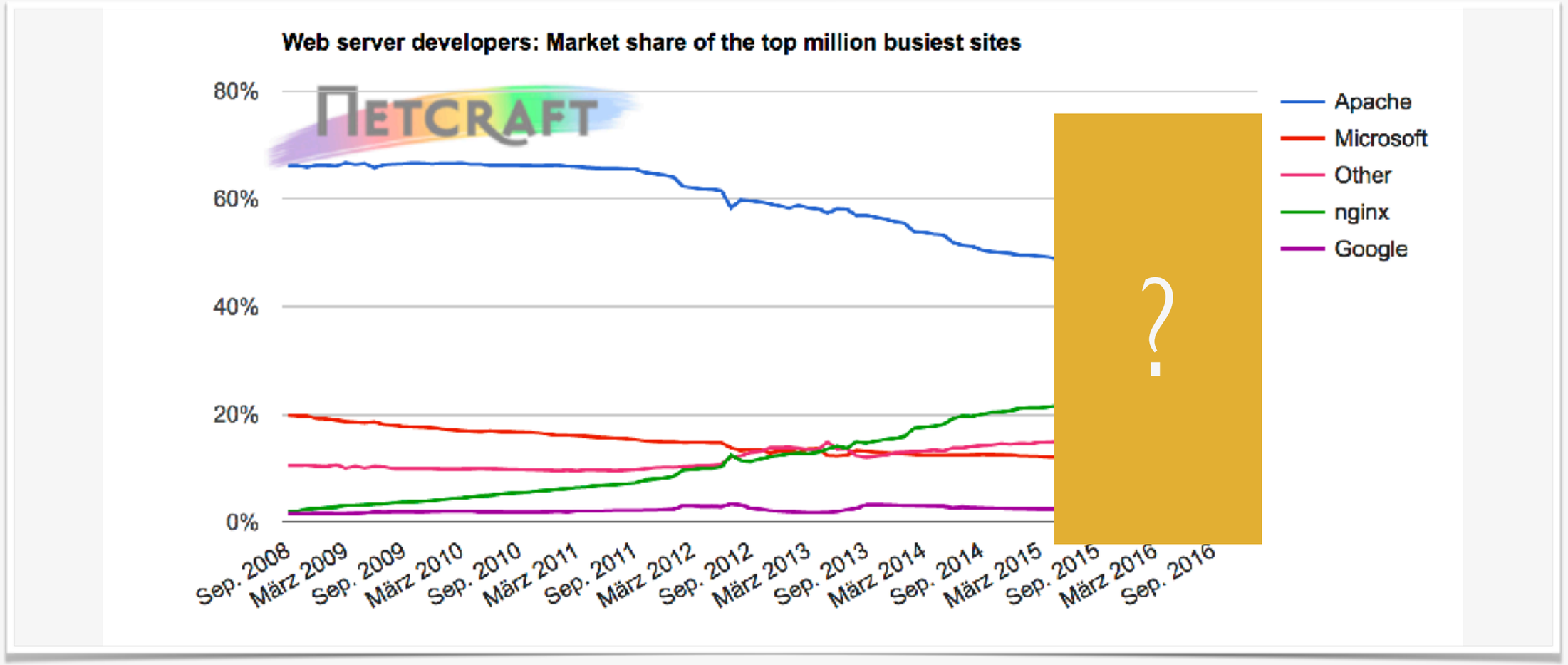
- Apache HTTP Server
- Microsoft IIS Server
- LiteSpeed Web Server
- Nginx
- Google Web Server

Webserver Marktanteile: Alle Seiten



trim/**netcraft**

Webserver Marktanteile: Top Million



tr.im/netcraft

HTTP

Abfragen gegen den Webserver

HTTP Protokoll

- HTTP Anfragen sind zustandslos, nach der Auslieferung der Antwort wird die Verbindung gelöst
- Der Client fragt einen URL (Uniform Resource Locator) an und der Webserver antwortet mit einem HTML-Dokument.
- Verweist der URL auf ein serverseitiges Skript, so wird dies zunächst ausgeführt und das generierte HTML-Dokument wird ausgeliefert.

HTTP Methoden

- GET
  - Informationen abfragen
- POST
  - Daten an den Server senden
- PUT
  - Element aktualisieren
- DELETE
  - Element löschen
- HEAD
  - Lesen der Dateiinformationen eines Dokuments
- OPTIONS
  - Angabe der erlaubten Methoden für diese URL

URL-Anfrage

steht für eine Anfrage nach dem Hypertext Transfer Protocol

steht für den Uniform Resource Locator (URL)

steht als Trennzeichen zwischen URL und Parametern

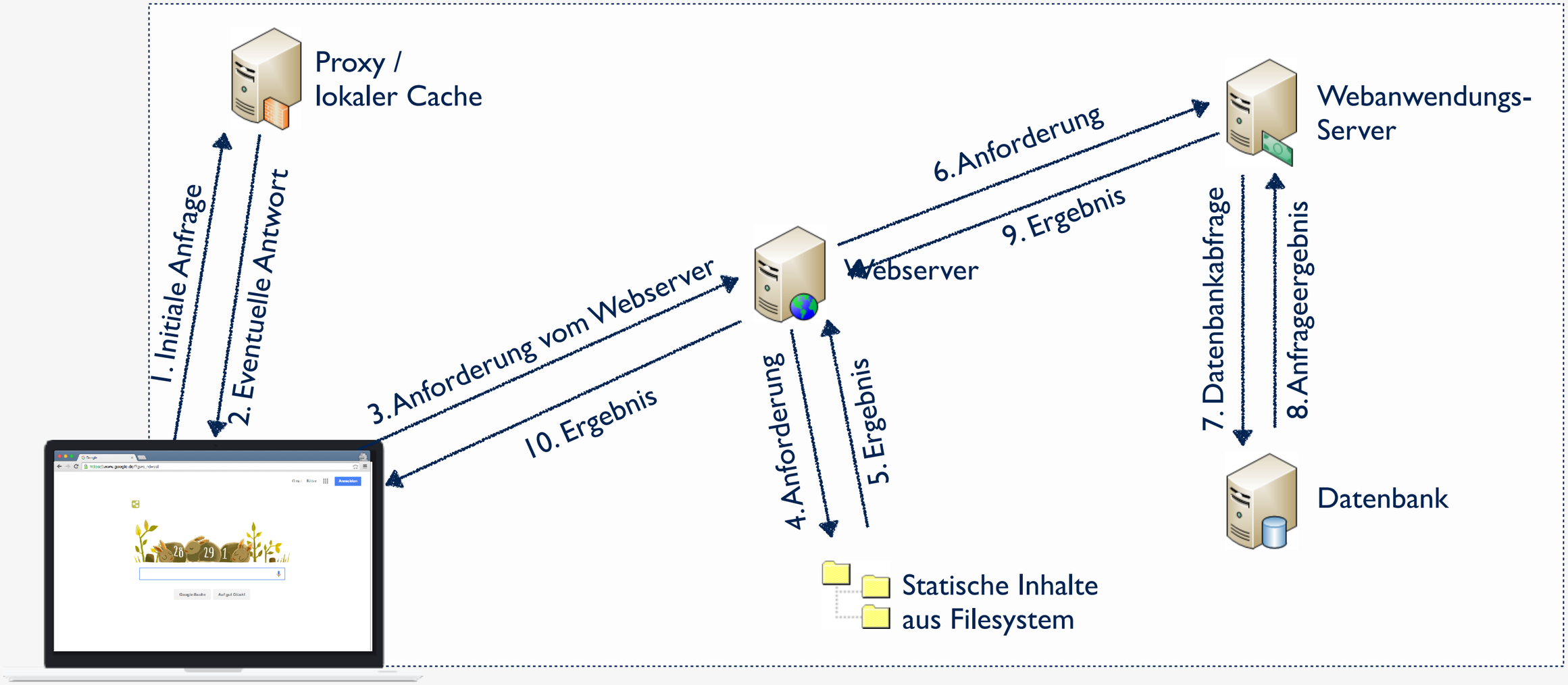
erster Parameter und Wert der Anfrage im Key-Value-Format

steht als Trennzeichen zwischen verschiedenen Parametern

zweites Key-Value Paar

Quelle: wikipedia.de

Schematischer Ablauf einer (dynamischen) Webanfrage



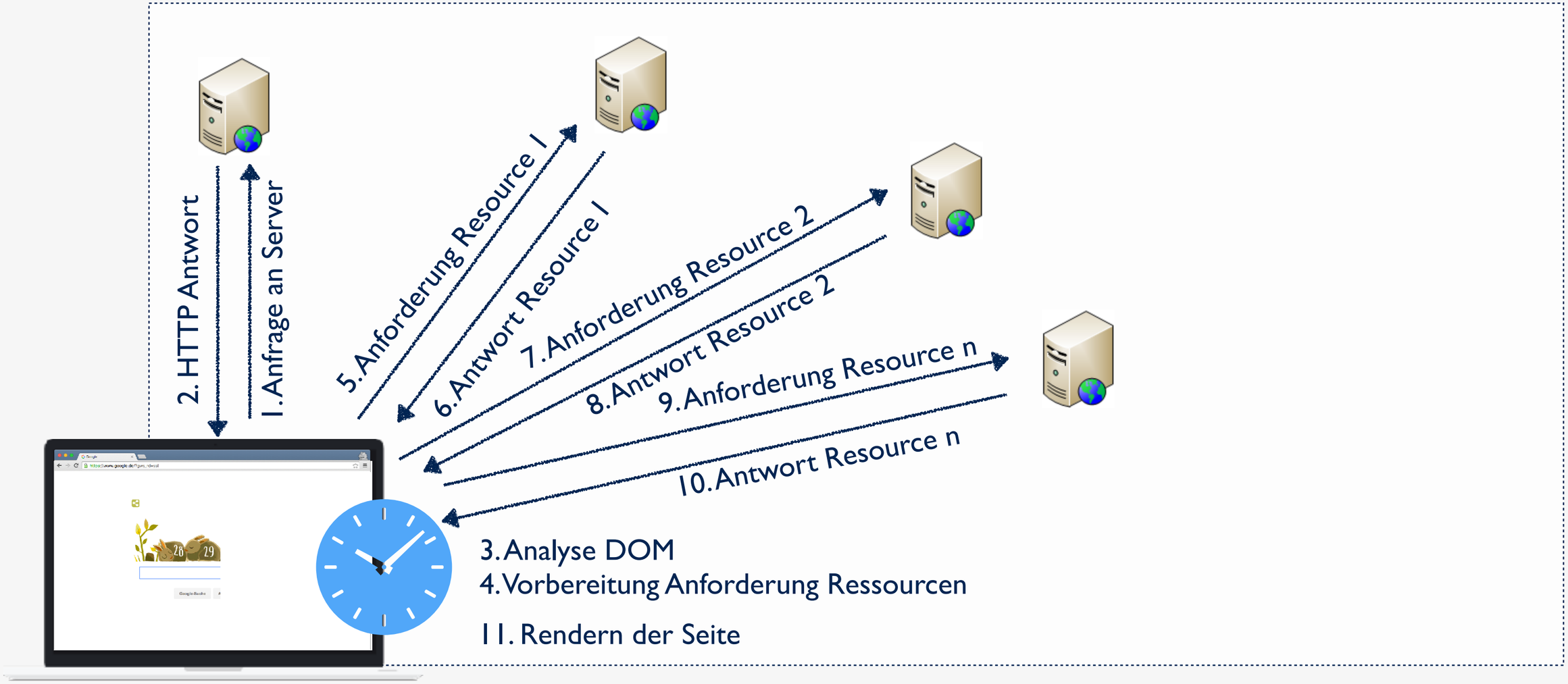
Sessionverwaltung

- HTTP ist ein zustandsloses Protokoll. Nach Bedienung der Anfrage durch den Server besteht keine Verbindung mehr zum Client.
- Gleichzeitig besteht aber die Notwendigkeit, einen Benutzer über mehrere Seiten hinweg zu verfolgen
  - Mehrstufige Formulare
  - Warenkorb beim Online-Shopping
  - Benutzer-Tracking für Werbetreibende und Social Networks
  - Webmailer/CMS/etc. mit sich öffnenden Fenstern zum Verfassen von Artikeln
- Varianten zur Realisierung des Session-Management
  - Hidden Fields in Formularen
  - URL Rewriting
  - SSL-Key der HTTPS-Verbindung
  - Cookies

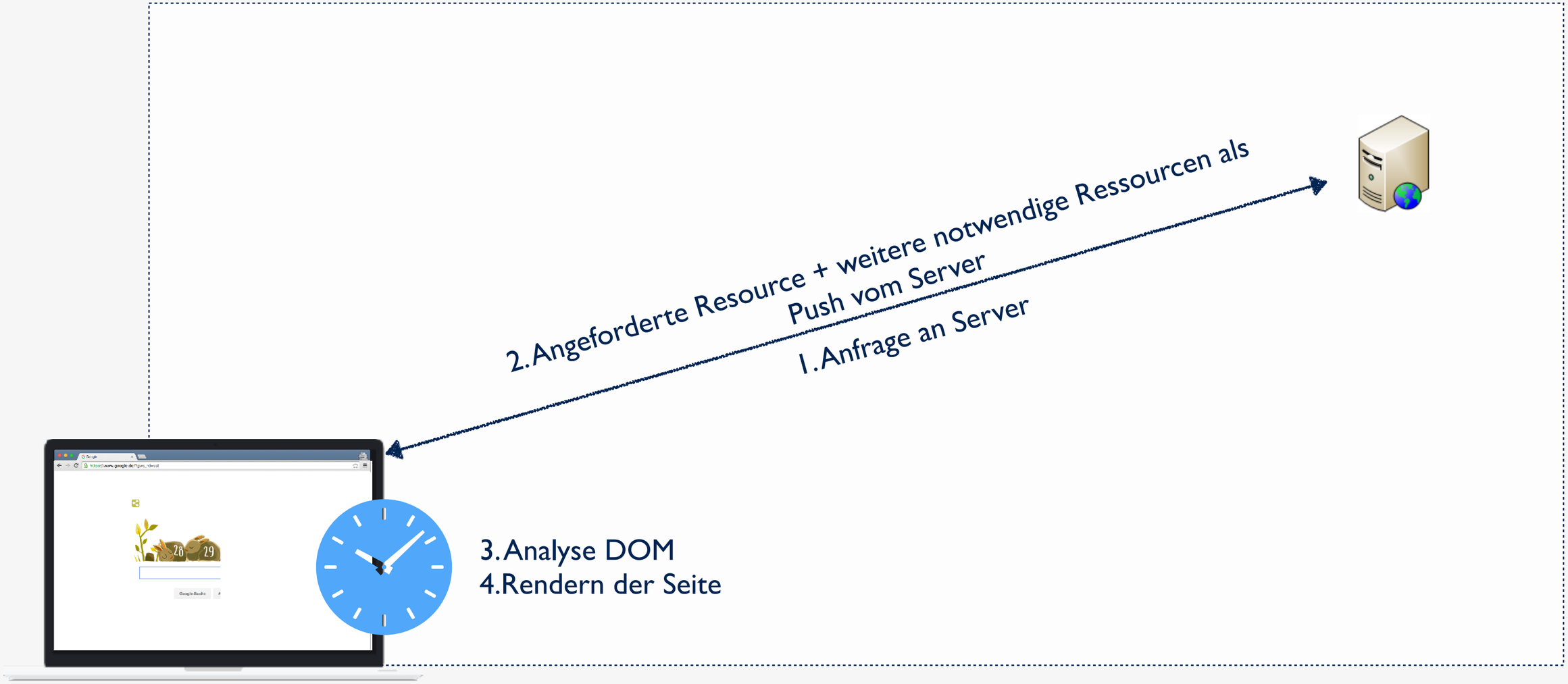
HTTP/2

- Nachfolger der Version 1.1 von 1996
- Aus dem Protokoll „SPDY“ von Google entstanden
- Bereits in allen verbreiteten Browsern und Webservern implementiert
- Zwischen Client und Server wird eine dauerhafte Verbindung aufgebaut und Daten werden über diese Verbindung ausgetauscht (Multiplexing)
- Entwicklung für aktuelle Anforderungen um folgende Probleme zu lösen
  - Performanceprobleme durch viele Dateien (CSS/JS/Images) beim Aufruf von Seiten
  - Große Header mit Session-Informationen
  - Nachladen der notwendigen Ressourcen nach Initialisierung des DOM
  - Fehlende durchgängige Kompression

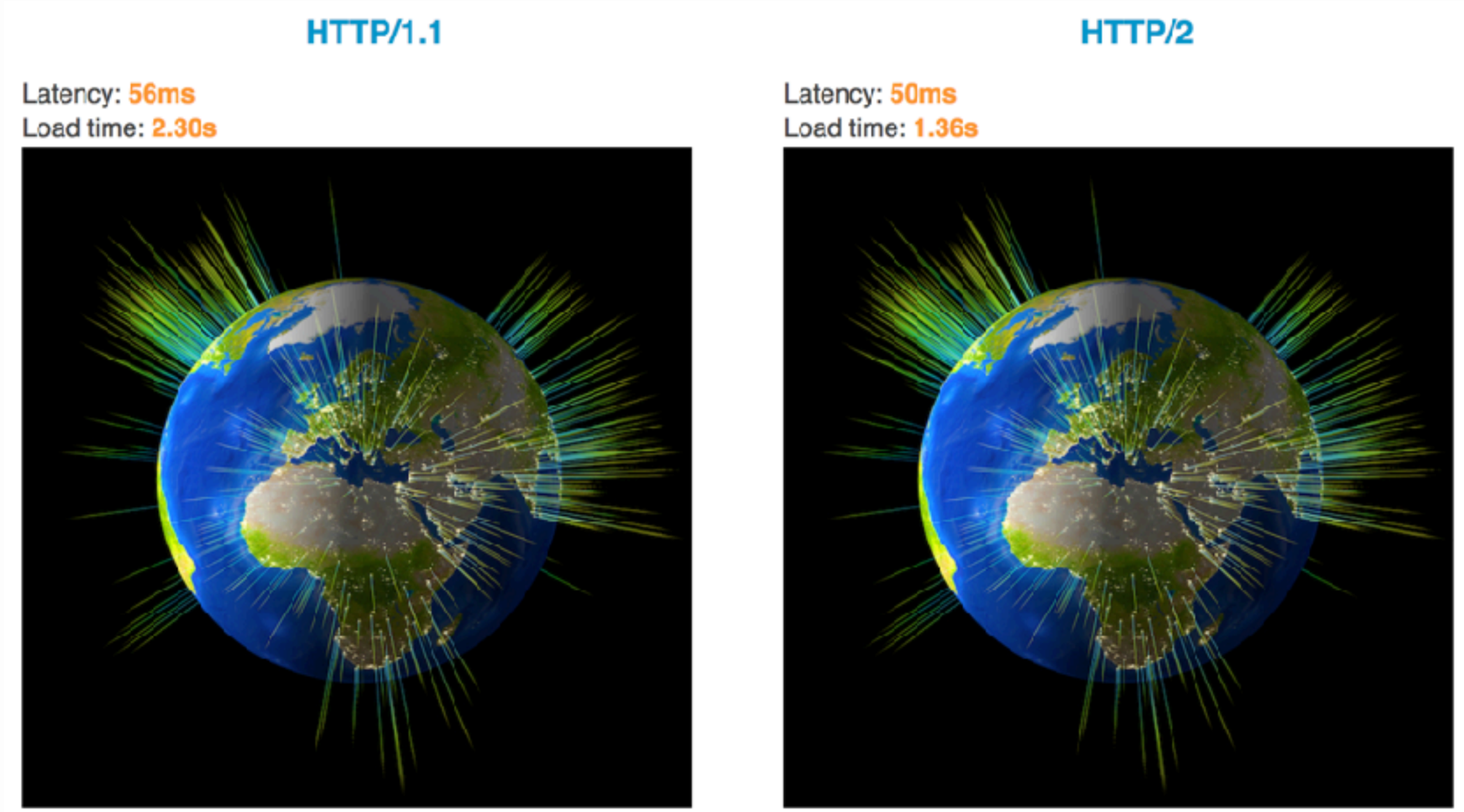
Anfrage HTTP/1.1



Anfrage HTTP/2



Aufbau HTTP/1.1 vs HTTP/2



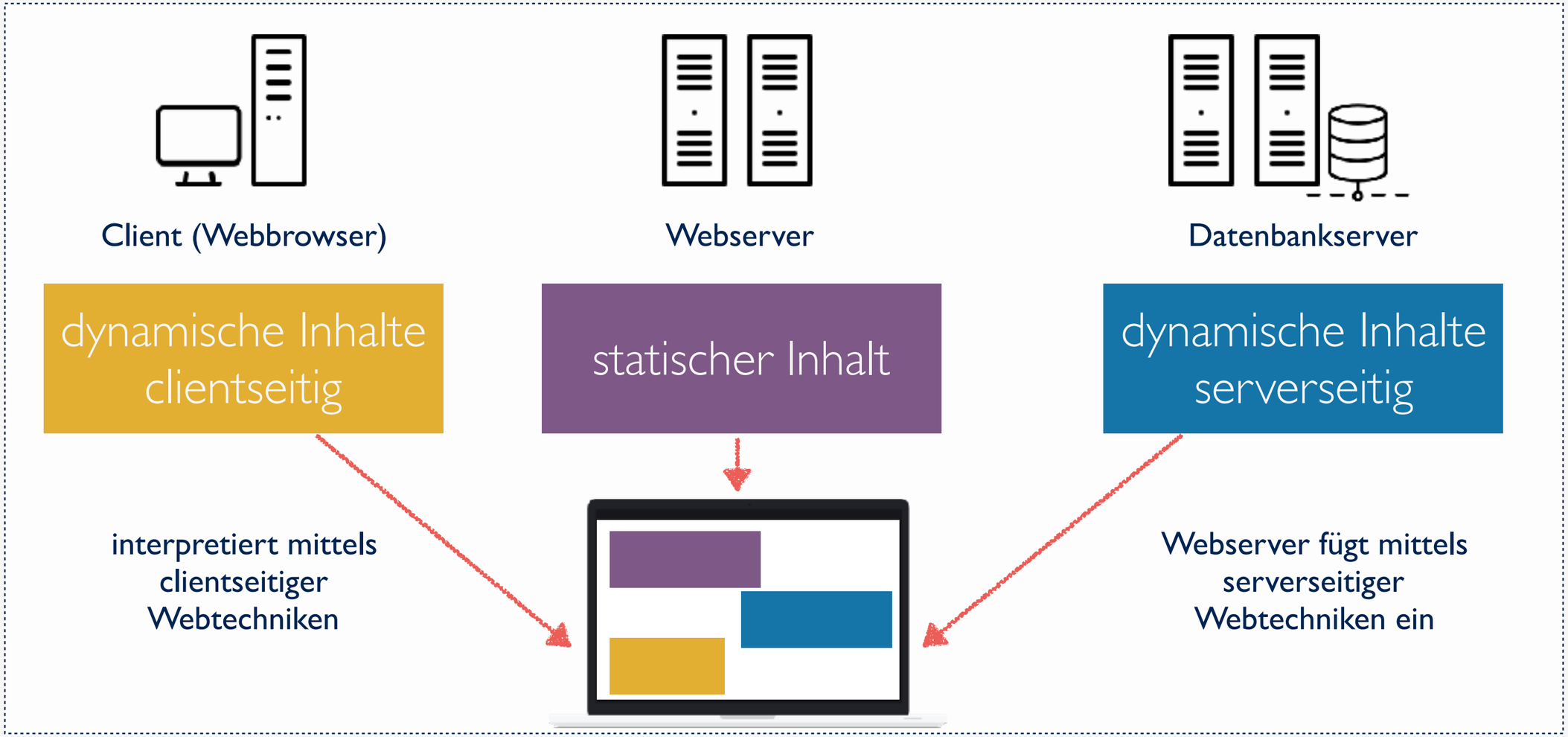
Serverseitige Programmierung

Anfragen dynamisch beantworten

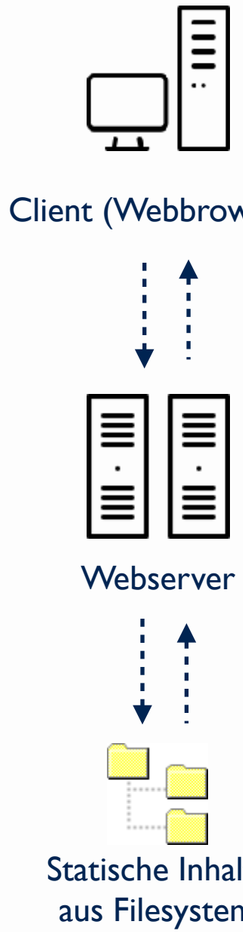
Serverseitige Programmierung

- 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

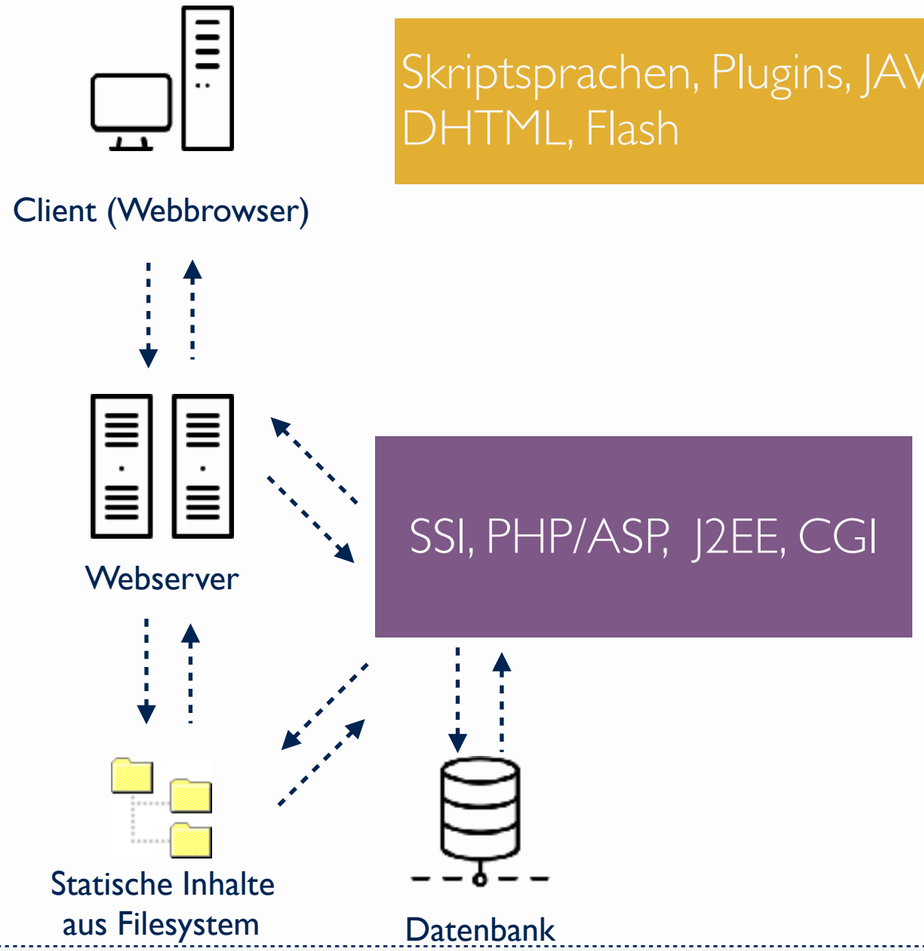


Anfrage an den Webserver
durch Hyperlink oder Direktaufruf

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

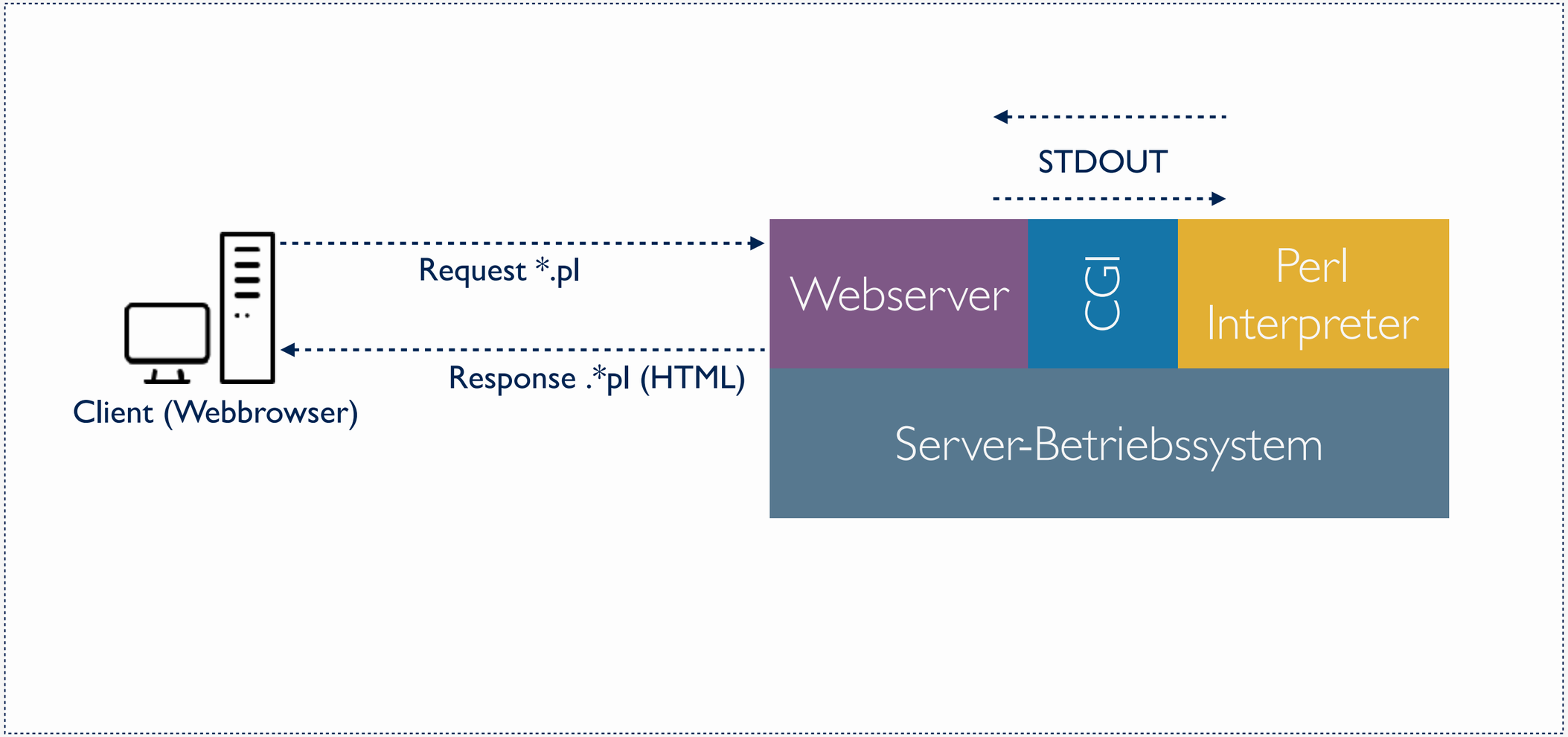
Serverseitige Programmierung: CGI

- 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

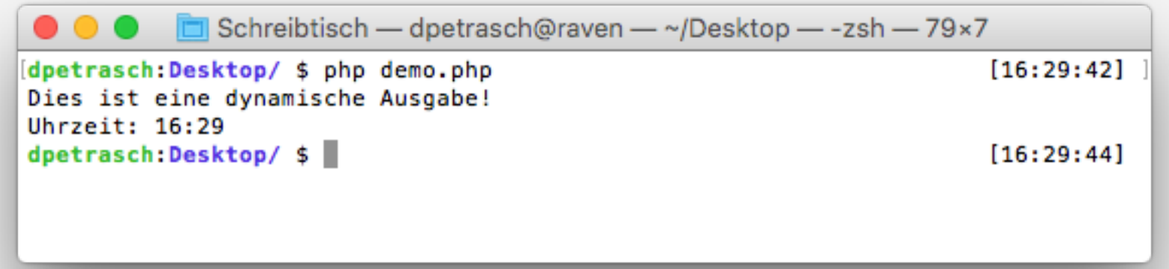
- Perl ist eine Skriptsprache, die zur Laufzeit interpretiert wird und daher als Interpretersprache keinen Compiler benötigt.
- Perl setzt nur einen auf dem Server installierten Perl-Interpreter voraus
- Perl hat Ähnlichkeiten zur C-Syntax und wurde ursprünglich für Unix-Plattformen entwickelt, um schnell 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



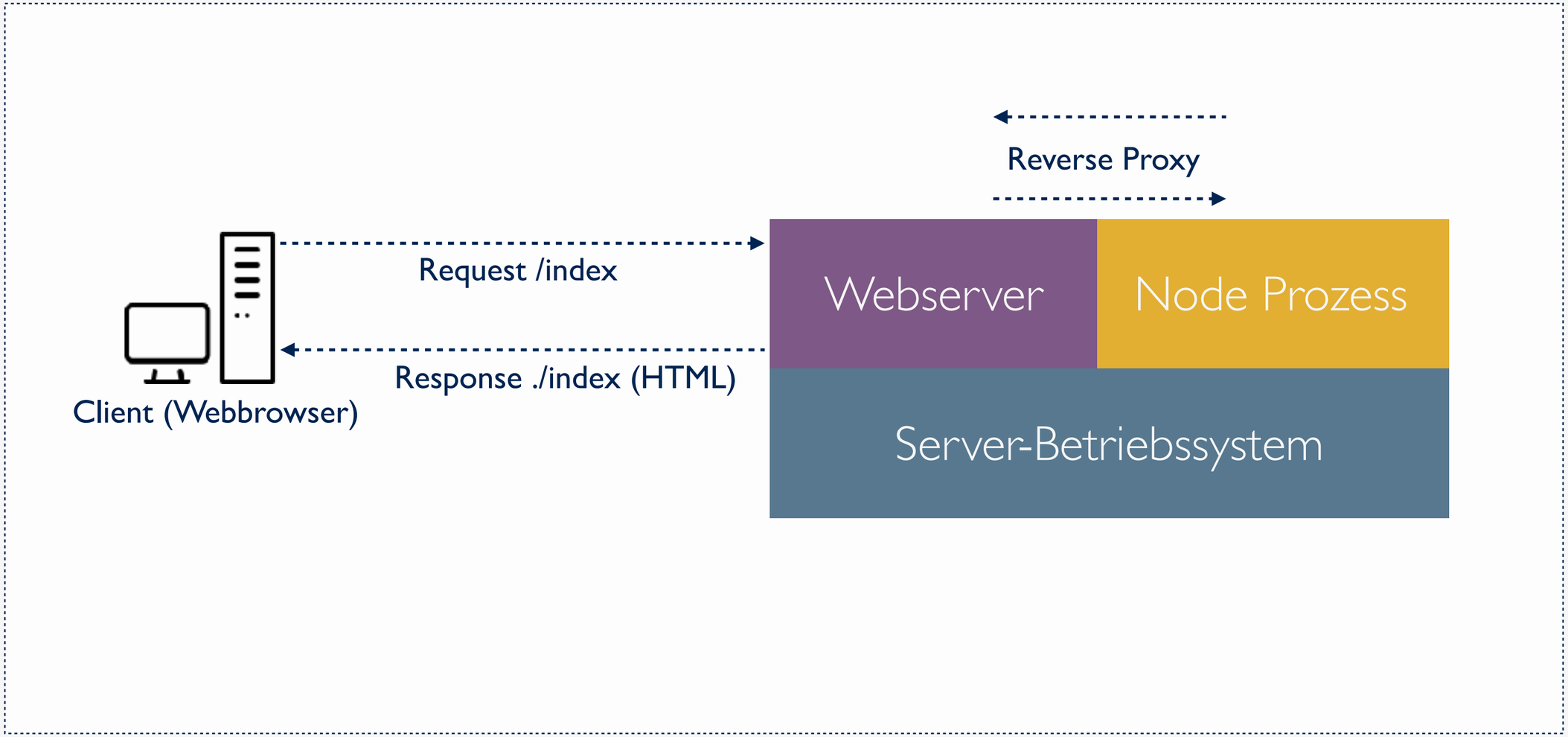

```php
<?php
    echo "Dies ist eine dynamische Ausgabe! \n";

    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

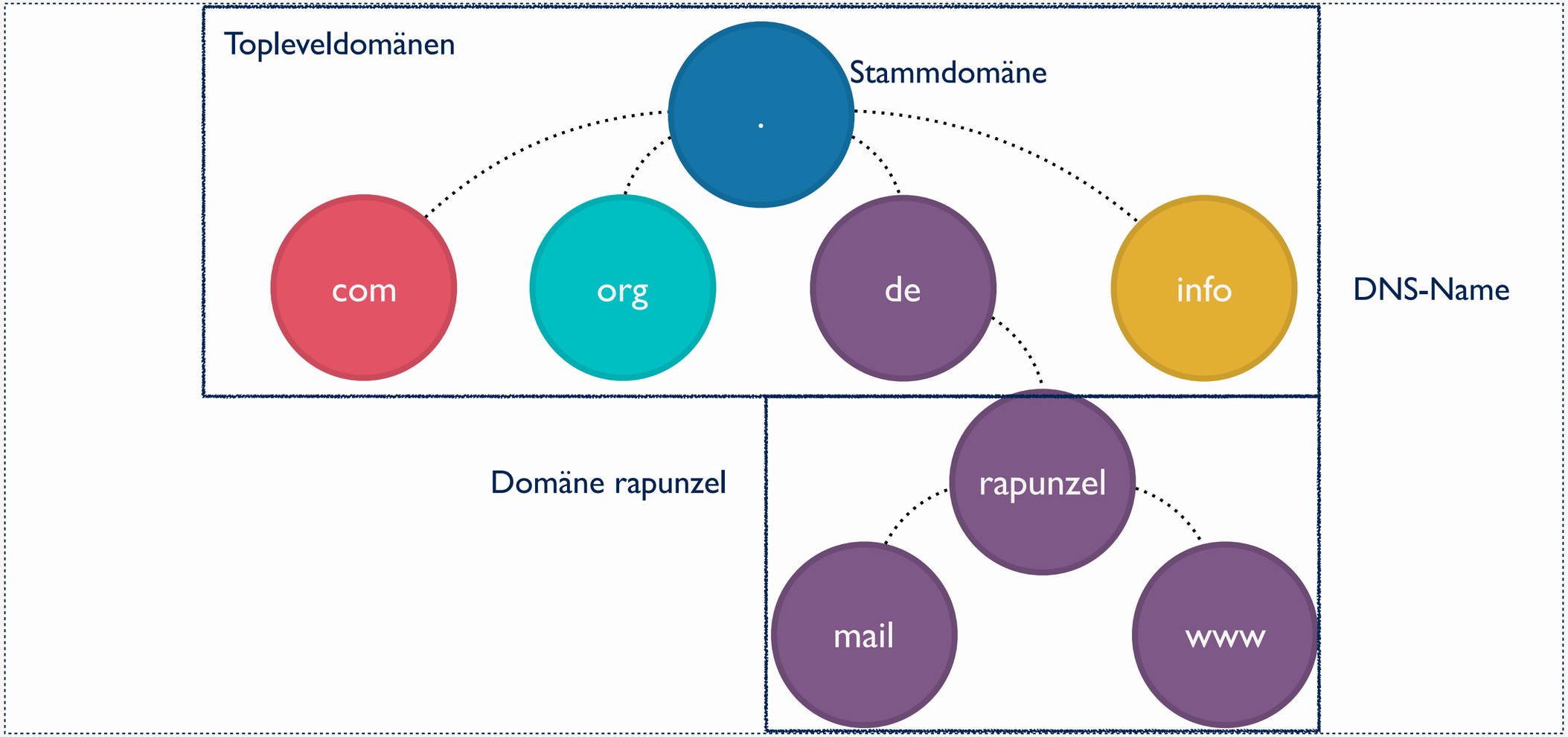
Funktionsweise von Node.js und einem Reverse-Proxy



DNS

Domain Name System

DNS Namespace



Quelle: wikipedia.de

Wichtige Begriffe

DNS Server	jeder Computer, der den DNS-Dienst ausführt, eine DNS-Datenbank hält und auf Clientanfragen antwortet
DNS resolvers	Programme, die mit Hilfe von DNS-Abfragen Informationen auf den Servern abfragen; Resolver können auf Clients oder Servern ausgeführt werden
Resource records	Informationen in der DNS-Datenbank, die für die Verarbeitung von Clientanfragen verwendet werden. DNS-Server halten Ressourcen-Einträge für den Anteil des Namespace, für den sie autorisiert sind.
Zones	fortlaufende Anteile des DNS-Namespace, für die der DNS-Server autorisiert ist. ein DNS-Server kann für eine oder mehrere Zonen autorisiert sein

Zonen

- Zonen
  - Zone = fortlaufender Anteil am DNS-Namespace
  - jede Zone ist an einem Domänenknoten verankert
  - Zone ≠ DNS-Domäne!
  - DNS-Domäne = Teilstruktur des DNS-Namespace
  - Zone = Teil des DNS-Namespace; kann mehrere DNS-Domänen enthalten
  - können von mehreren Servern verwaltet werden
  - Unterscheidung: primäre Zone / sekundäre Zone (Kopie)
  - Speicherung in Textfiles, AD-Integration möglich

DNS Server

- speichert Informationen über Zonen
- bedient Anfragen von Clients
  - nutzt eigene Zonen
  - nutzt Cache (auch negativen Cache)
  - fragt andere Server
  - leitet weiter
- kann gleichzeitig primärer Server und sekundärer Server einer anderen Zone sein
- nimmt Änderungen nur für primäre Zonen entgegen

Vorteile von Sekundarservern

- Fehlertoleranz
- Reduktion der Netzwerklast
- Reduktion der Datenlast auf Primärserver
- Caching-only Server zur weiteren Performance-Steigerung

Forwarders und Slaves

- bei nichterfolgreicher Abfrage der eigenen Zone ist Kontaktierung anderer DNS-Server möglich
- dadurch unter Umständen hohe Netzlast
- Vermeidung durch Definition von Weiterleitungsservern, die weitere Abfragen durchführen
- dadurch Zentralisierung der Abfragen für fremde Zonen => bessere Cacheausnutzung
- nicht exklusiver Modus = Forwarder
- exklusiver Modus = Slave

  dieser versucht auch nach fehlgeschlagener Weiterleitung keine eigenen zusätzlichen DNS-Abfrage

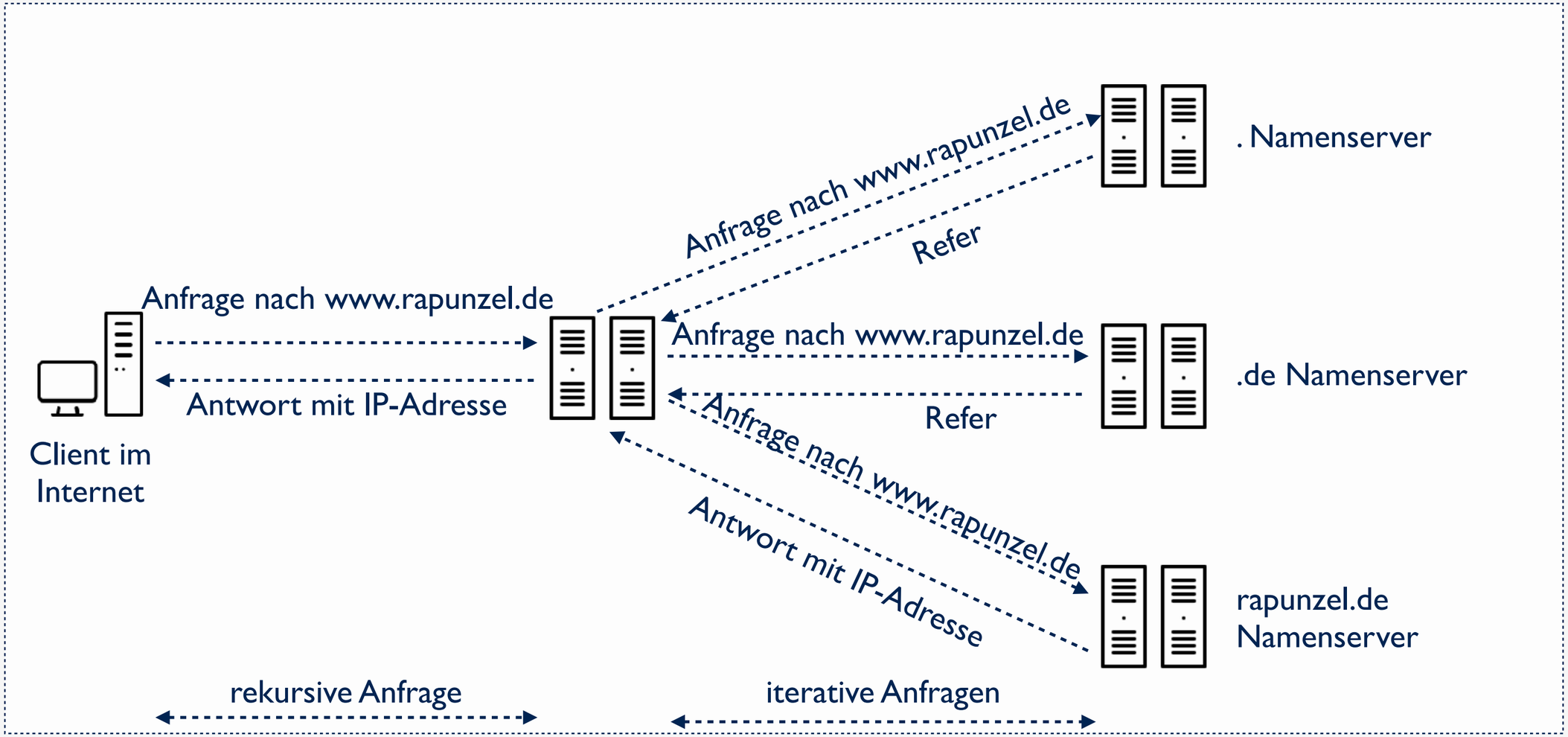
Lastaufteilung

- DNS-Server können einfache Lastaufteilung mittels Round-Robin-Verfahren durchführen
- einem Namenseintrag sind mehrere IP-Adressen zugeordnet
- bei Anfragen werden die Adressen in rotierender Folge ausgegeben
- sinnvoll z.B. bei WWW – Servern
- keine wirkliche Lastaufteilung, aber eine einfache Methode um Netzwerkressourcen besser auszunutzen

Rekursive und iterative Anfragen

- Rekursive Anfragen
  - grundsätzlicher Anfragetyp von DNS-Clients
  - Client erwartet bei rekursiver Anfrage eine endgültige Antwort des DNS-Servers
  - hat dieser keine Informationen muss er zur Beantwortung einer rekursiven Clientanfrage selbst weitere Anfragen stellen
- Iterative Anfragen
  - Client erlaubt dem Server statt einer endgültigen Antwort auch eine bestmögliche Antwort zu geben, z.B. den wahrscheinlich zuständigen DNS-Server
  - DNS-Server fragt dann bei diesem wieder iterativ nach
  - durchläuft auf diese Weise den DNS-Baum - „walking the tree“
  - üblicher Anfragetyp zwischen DNS-Servern

Einführung in DNS: Beispiel



Resource Records

Owner (Besitzer)	Name des Hosts oder der DNS-Domäne, zu dem/der der Eintrag gehört
TTL	32-Bit Integer, der die Gültigkeitsdauer angibt (optional)
Class	Protokollfamilie, immer „IN“
Type	Typ des Ressourceneintrags
RDATA	Ressourceneintragsdaten, variabler Typ, je nach Type des Eintrags

SOA Resource Record (ursprung einer Zone)

- RDATA enthält:
  - authorized server = primärer Server der Zone
  - responsible person (Achtung: kein „@“, sondern „.“ nutzen)
  - serial number (Anzahl der Aktualisierungen der Zone)
  - refresh (wie oft ist die Zone zu replizieren)
  - retry (Timeout für Zonentransfer)
  - expire (Verfall der Zone, wenn Zonentransfer nicht erfolgreich)
  - minimum TTL (Default-TTL-Wert für alle Einträge)

```
keil-it.de.   IN SOA (
              dc1.rapunzel.de        ; autorisierter Server
              dnsadmin.rapunzel.de   ; Zonenadministrator
              1441                   ; Seriennummer
              3600                   ; Refresh (1h)
              600                    ; Retry (10min)
              86400                  ; Expire (1d)
              60     )               ; minimum TTL (1min)
```

/dpetrasch/DHBW

NS Resource Record

- geben die für die Zone autorisierten Namensserver an
- jede Zone muss mindestens einen NS-Eintrag enthalten

```
demo.rapunzel.de.    IN    NS    dc1.demo.rapunzel.de
```

/dpetrasch/DHBW

A Resource Record

- Address Resource Record ordnet einem FQDN eine IP zu

```
demo.rapunzel.de.    IN    A    127.22.23.4
```

/dpetrasch/DHBW

PTR Resource Record

- Pointer Resource Record ordnet einer IP den FQDN zu
- rückwärts geschriebener IP & „in-addr.arpa.“

```
4.23.22.172.in-addr.arpa.    IN    PTR    demo.rapunzel.de
```

/dpetrasch/DHBW

CNAME Resource Record

- Canonial Name resource record erstellt einen Aliasnamen
- ein FQDN kann durch mehrere CNAMEs adressiert werden
- ermöglicht Zugriff über „freundliche Namen“

```
www.rapunzel.de.    IN    CNAME    demo.rapunzel.de
```

/dpetrasch/DHBW

MX Resource Record

- Mail Exchange record legt Mailaustauschserver fest
- mehrere Einträge können mit versch. Priorität angelegt werde
- niedrigster Wert bedeutet größte Präferenz

```
*.rapunzel.de   IN   MX    10   mail1.rapunzel.de
*.rapunzel.de   IN   MX    20   mail2.rapunzel.de
```

/dpetrasch/DHBW

SRV Resource Record

- Service Resource record ermöglichen Standortbestimmung bestimmter Dienste
- für die AD-Struktur von extrem hoher Bedeutung

```
_http._tcp.rapunzel.de  IN  SRC  0  0  80  demo1.rapunzel.de

_Service._Proto.Name TTL Class SRV Priority Weight Port Target
```

/dpetrasch/DHBW

Zonen

- Forward Lookup Zonen
  - enthalten Informationen, wie Namensanfragen in IP-Adressen aufgelöst werden
  - es können alle Resource records außer PTR auftreten
  - müssen immer einen SOA und mindestens einen NS Eintrag enthalten
- Reverse Lookup Zonen
  - enthalten Informationen, wie IPs zu Namen aufgelöst werden
  - stellen innerhalb des DNS-Namespaces die Sonderdomäne in-addr.arpa dar
  - können nur SOA, NS, PTR und CNAME enthalten
  - im Regelfall zu jeder FLZ auch eine RLZ

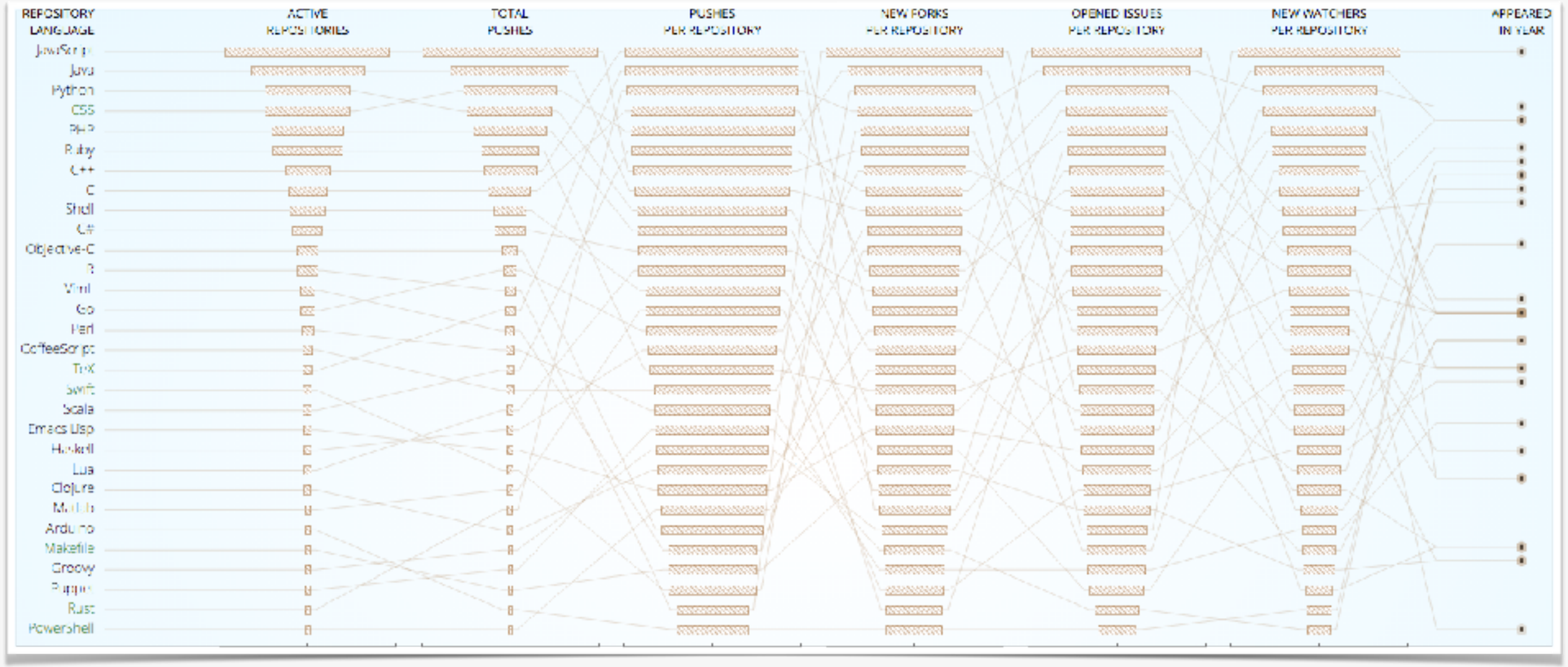
DynDNS

- Dynamisches DNS sind DNS-Einträge mit wechselnden IP-Adressen (bspw. im heimischen Netzwerk)
- Da sich die IP-Adresse in der Regel alle 24 Stunden ändert, muss auch das DNS aktualisiert werden
- Die DNS Einträge dürfen nicht lange zwischengespeichert werden, damit eine schnelle Änderung möglich ist. Hier wird das TTL meist auf 1 Minute gesetzt
- Eine Software oder ein Netzwerkgerät informiert bei einer Änderung der IP-Adresse den entsprechenden Nameserver eines DDNS-Anbieters, welcher den DNS-Eintrag aktualisiert
- Anbieter sind unter anderem:
  - DynDNS.org
  - NO-IP.com
  - Hersteller-eigene Dienste z.B. von NAS-Systemen

Übersicht der Websprachen

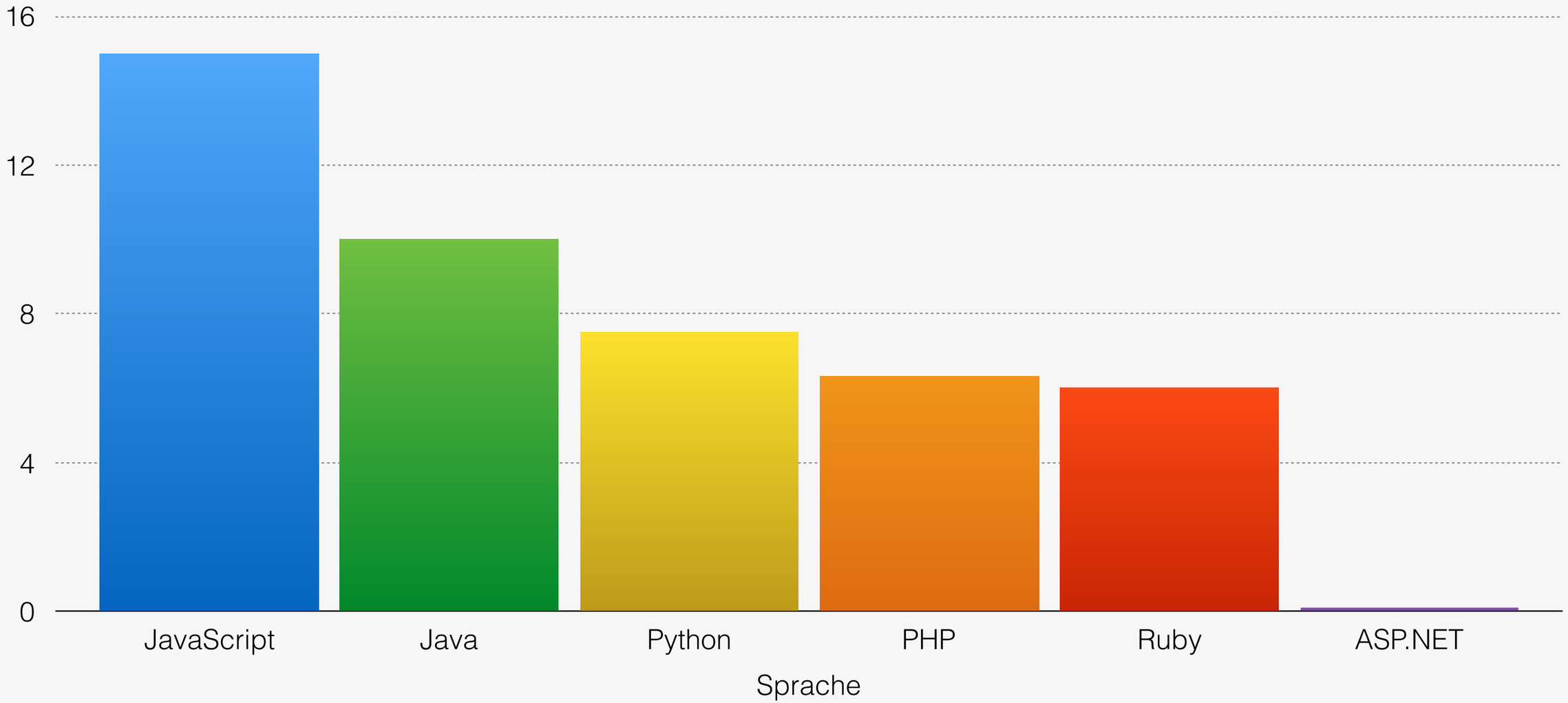
Aktuelle Sprachen und deren Verteilung

Übersicht über aktuelle GitHub Repositories



tr.im/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

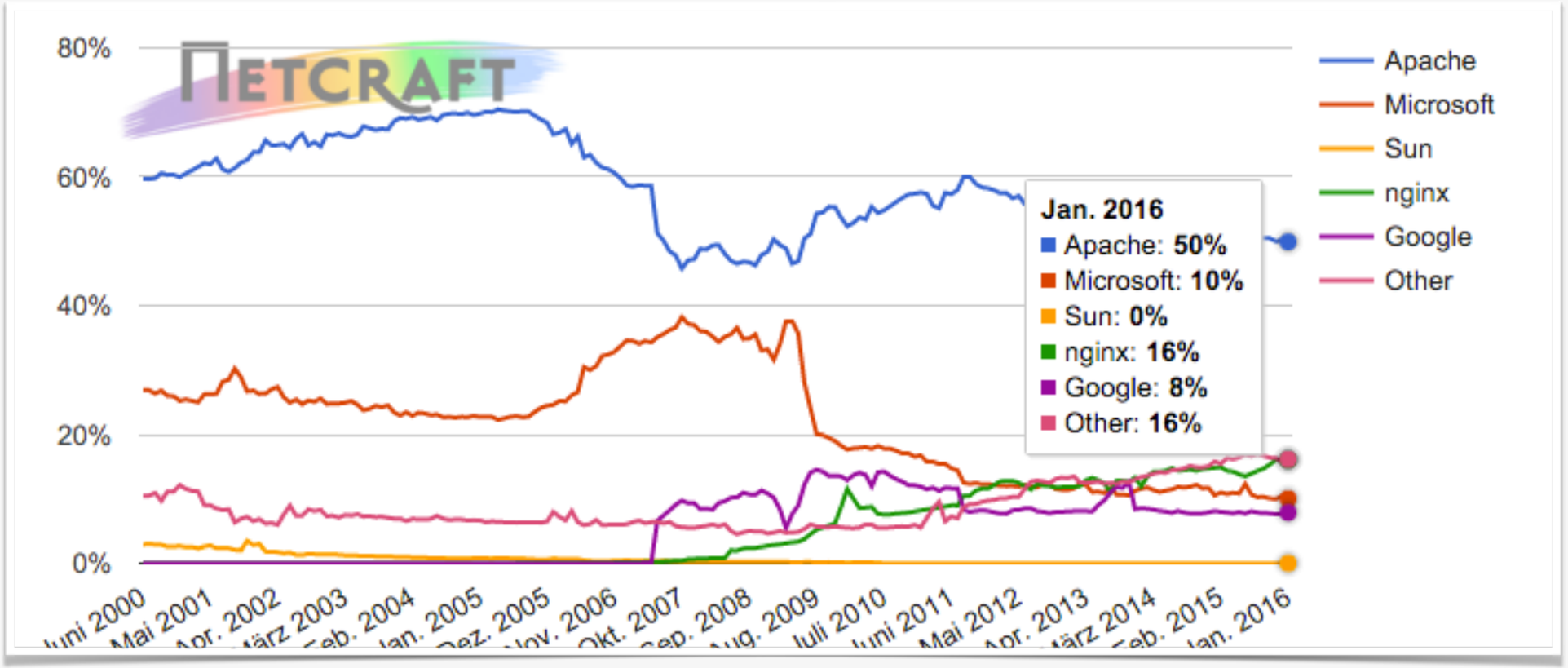
ASP.NET

- 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



trim/**netcraft**

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");
}
```

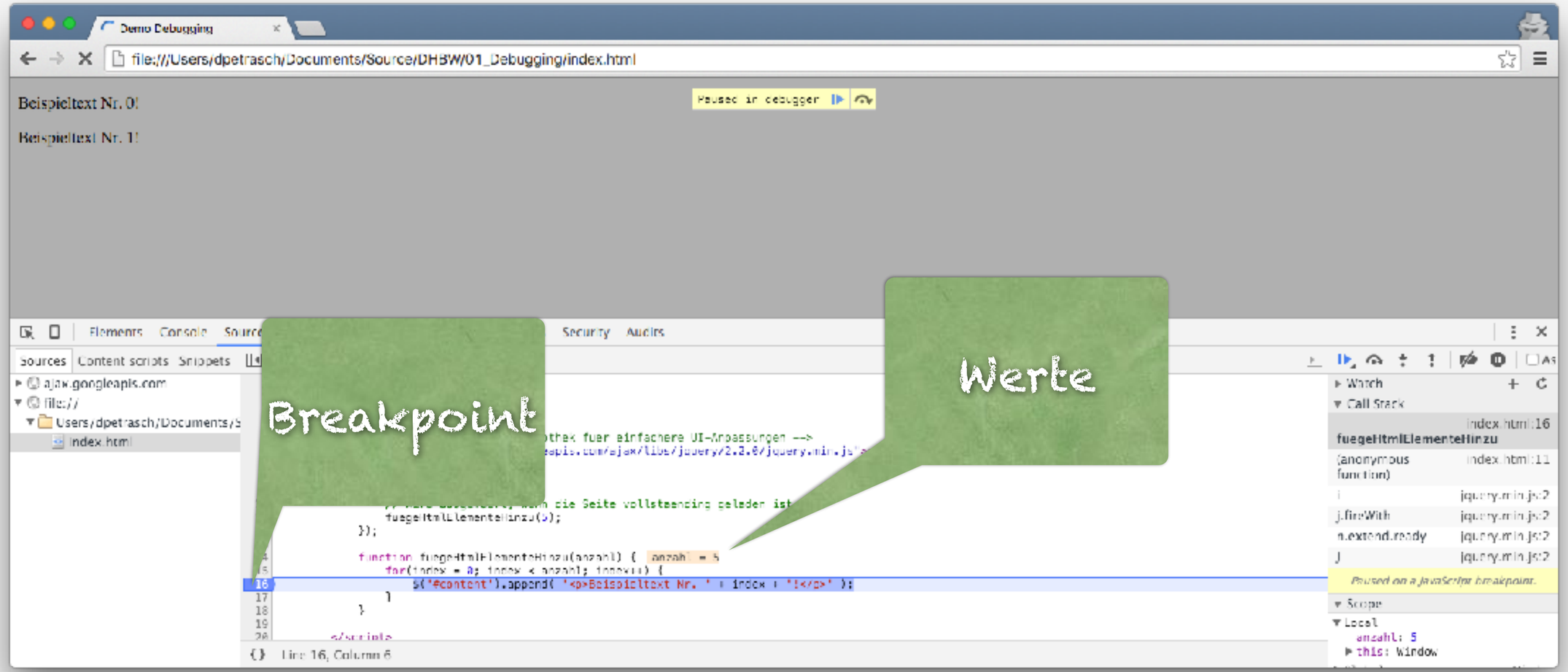
/dpetrasch/DHBW

Debuggen von Javascript-Quellcode

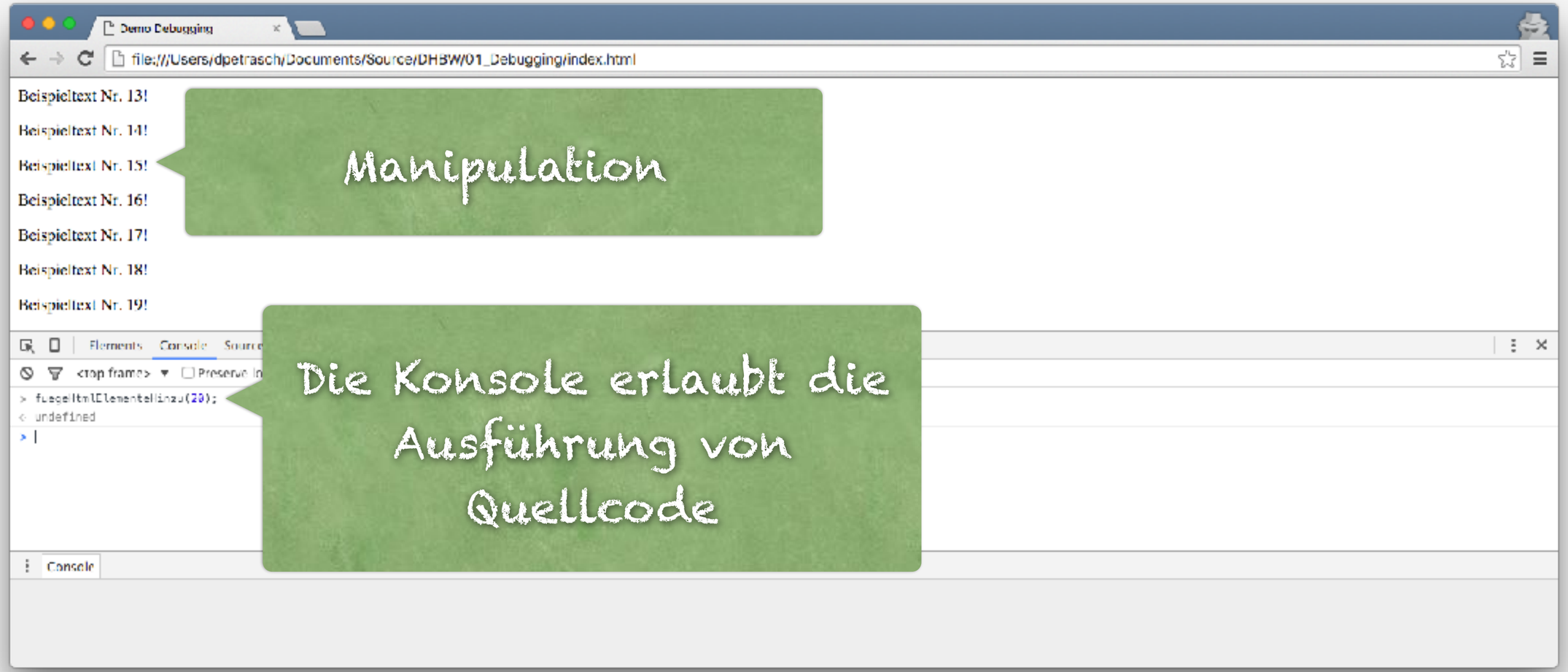
- Javascript-Quellcode, der beim Client ausgeführt wird, wird direkt im Browser ausgeführt und kann „debuggt“ werden - also angehalten und manipuliert werden
- Bei der Entwicklung und das Debuggen wichtig, um Fehler im Quellcode aufzuspüren und beheben zu können
- Die Konsole im Browser erlaubt es Entwicklern,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

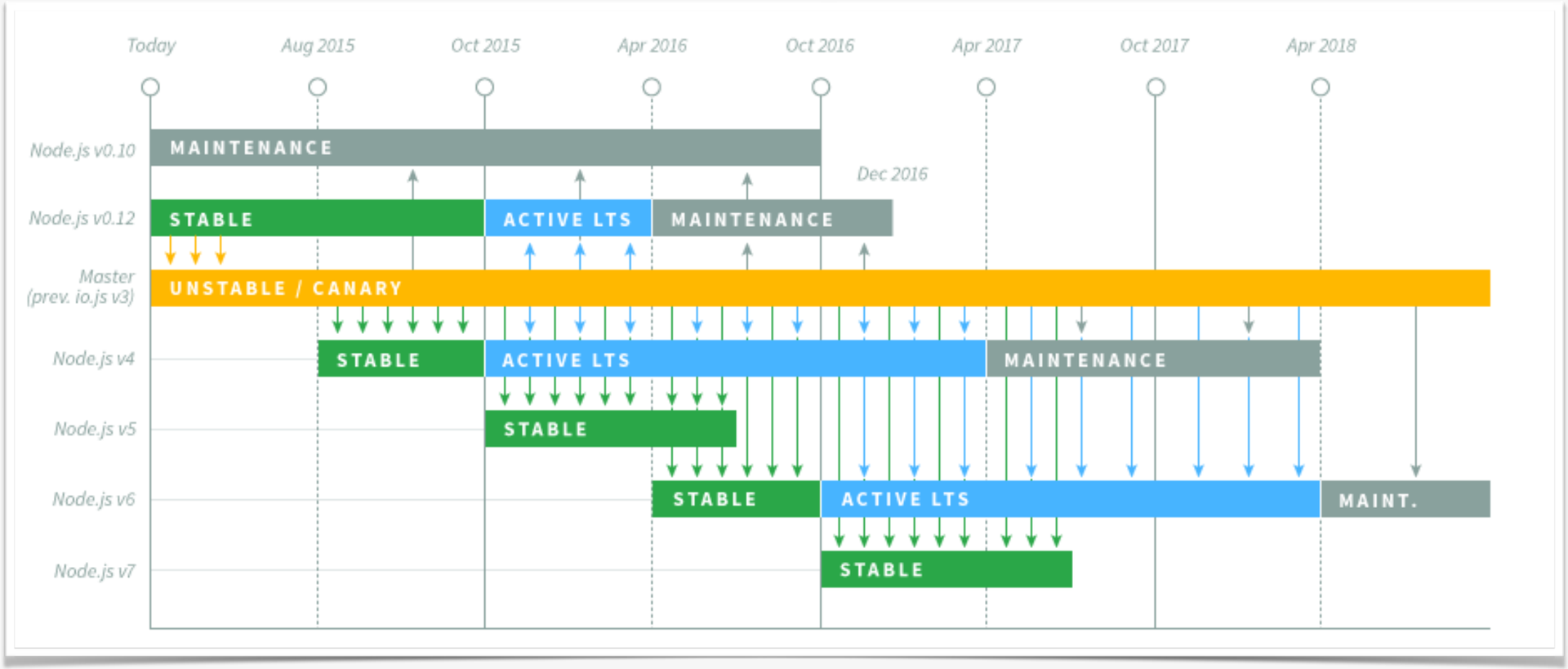
Einführung in Node.js

Allgemeines, Aufbau und Komponenten

Allgemeines

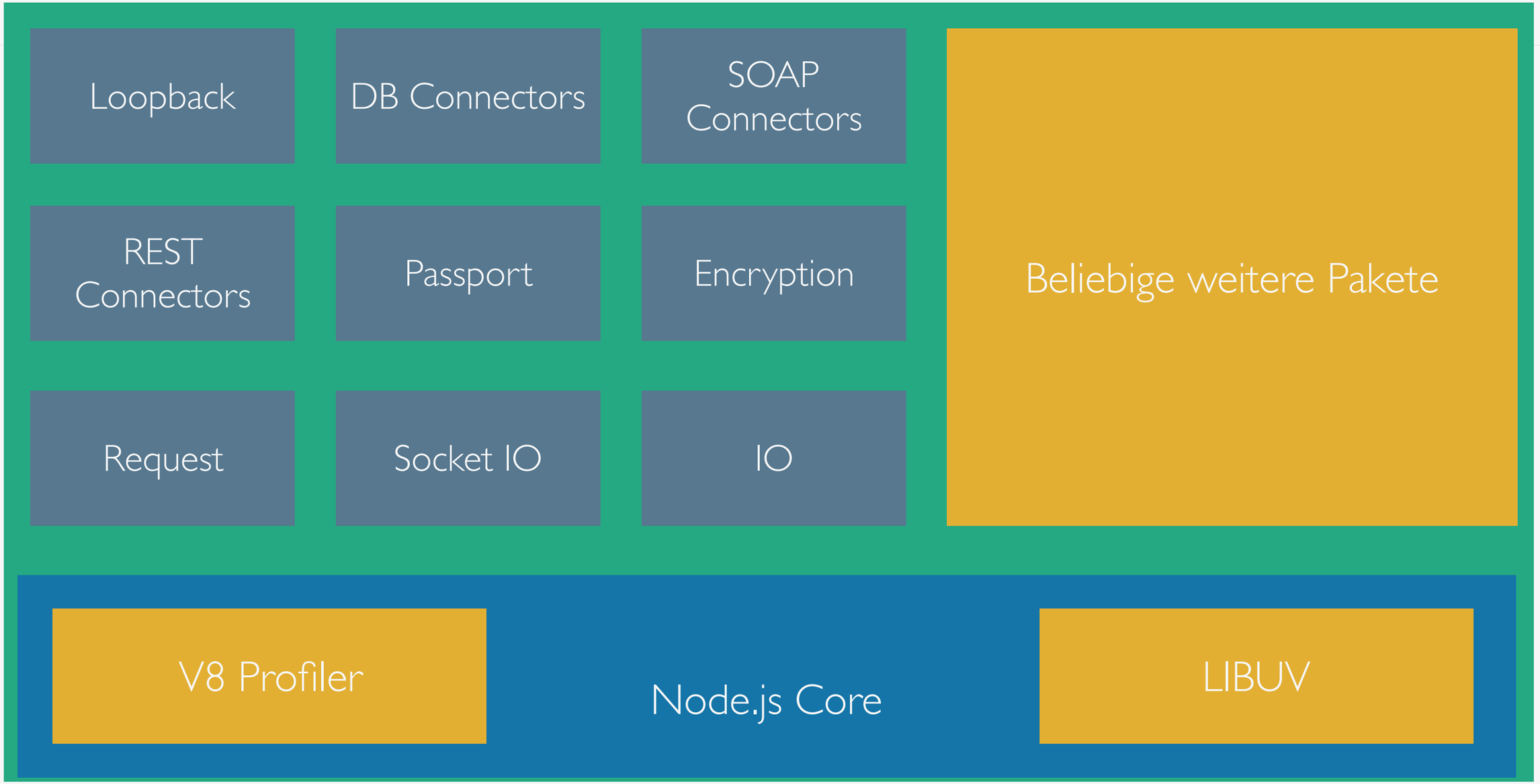
- Node.js ist eine serverseitige Plattform zum Betrieb von Netzwerkanwendungen
- Basiert auf der Google V8-Javascript-Engine
- Erweiterbar ist Node mit Paketen (Modulen)
- Es existiert ein Paketmanager (NPM)
- Das „Grundpaket“ enthält bereits viele APIs und Funktionen
- Es kann auch C++-Quellcode ausgeführt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

Versionen und Long Term Support



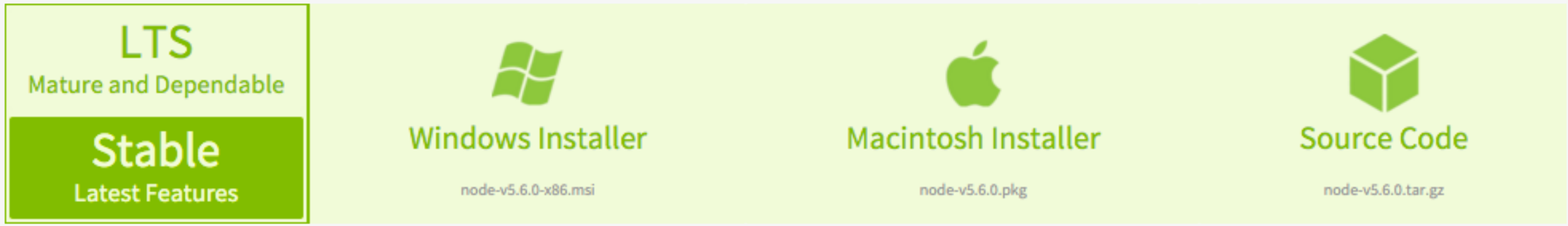
tr.im/nodelts

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

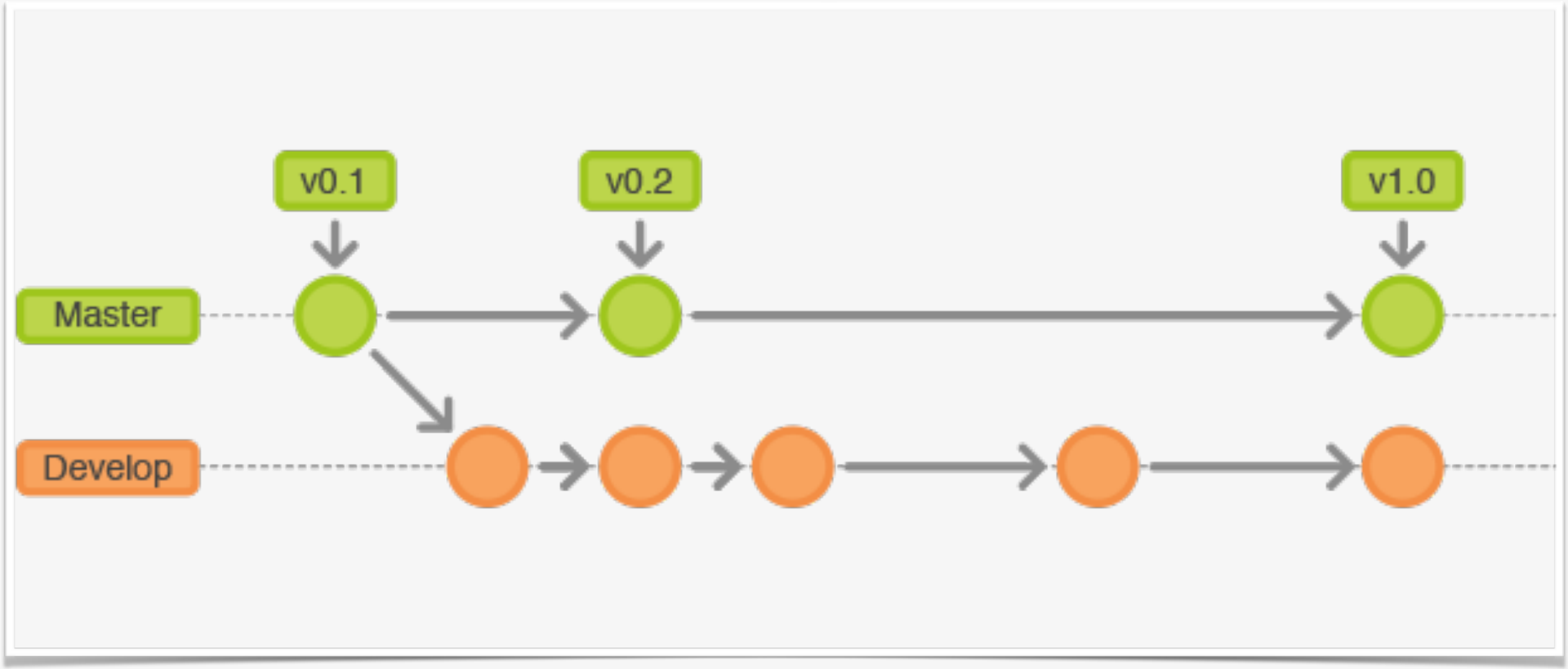
Vokabular: Git

- 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

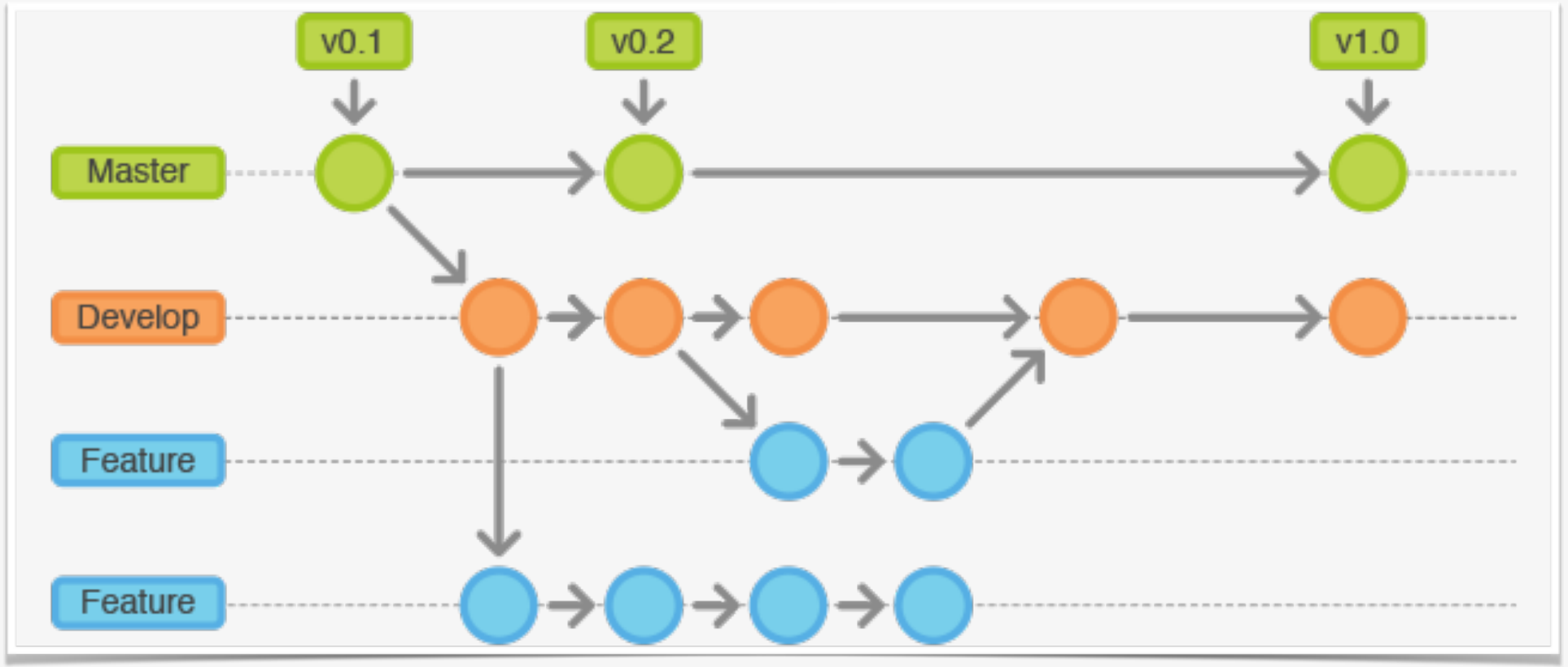
- Revert
  - Es kann zu einem bestimmten Commit in der Zeitlinie gesprungen werden. Die Änderungen an den Dateien wird dann bis zu dem Zeitpunkt rückgängig gemacht
- Init
  - Der Prozess zum Erstellen eines neuen Repositories wird „init“ genannt
- Checkout
  - Ein vorhandenes Repository auf dem System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

Git Workflow: Master und Entwicklung



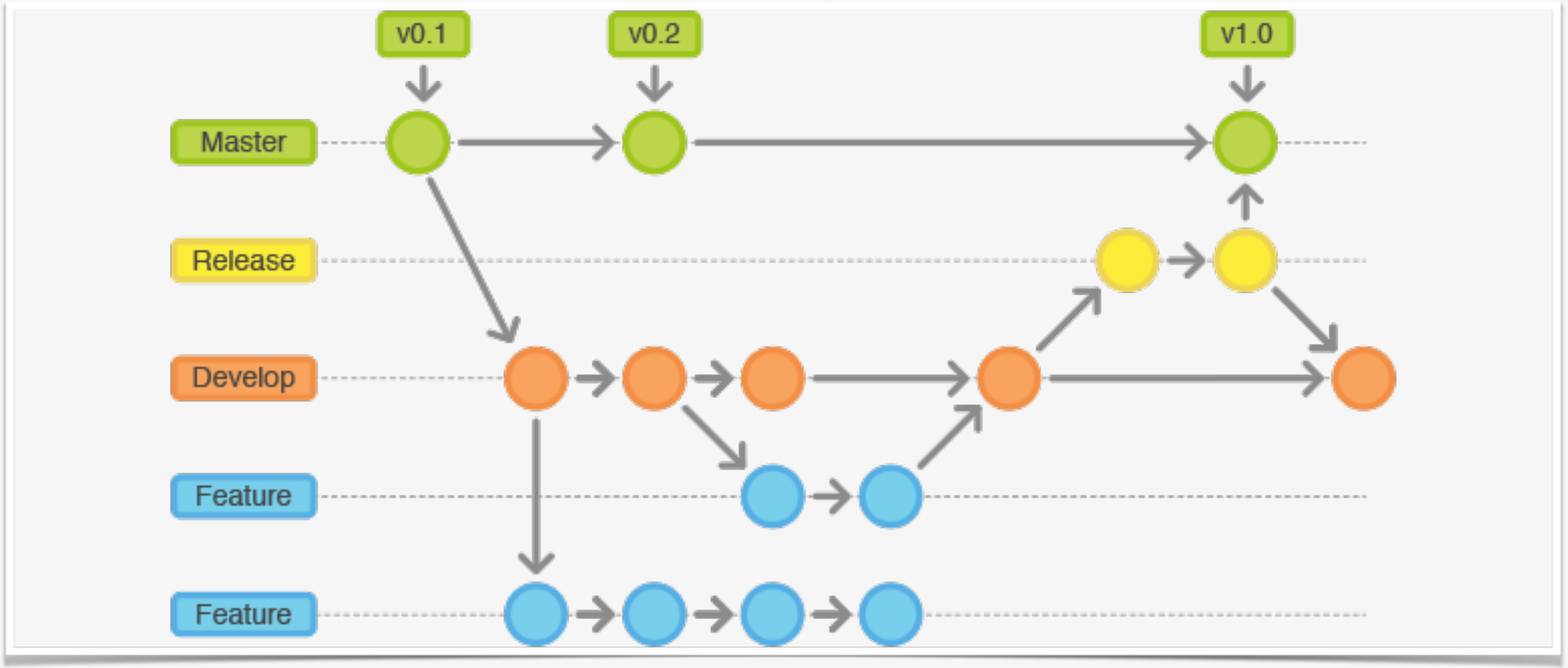
tr.im/git

Git Workflow: Funktionsbranches



tr.im/git

Git Workflow: Funktionsbranchen mit Release



trim/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