Basics Web-Technologies

Prof. Dr. Tobias Eggendorfer

Technische Hochschule Ingolstadt

Basics

Repetition DNS

History

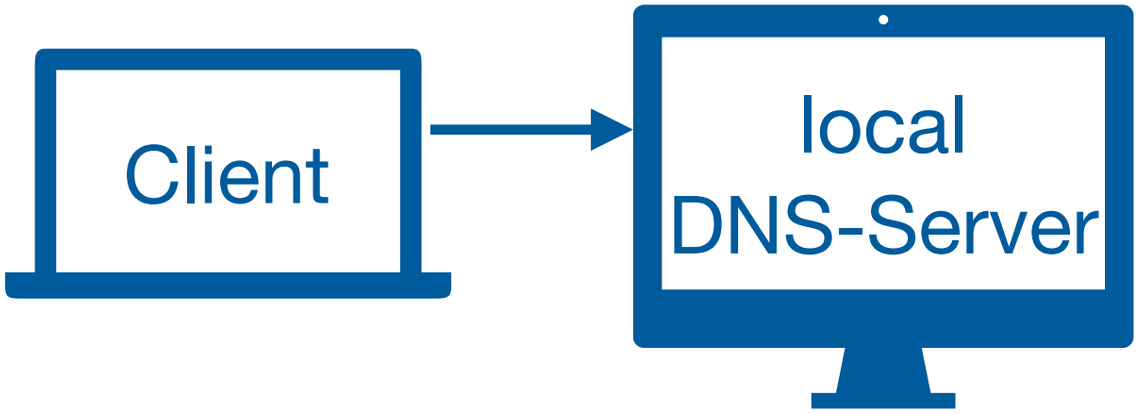
- Simplify memorizing systems
- first: /etc/hosts
- then: DNS
  - Distributed
  - Hierarchical

Naming concept

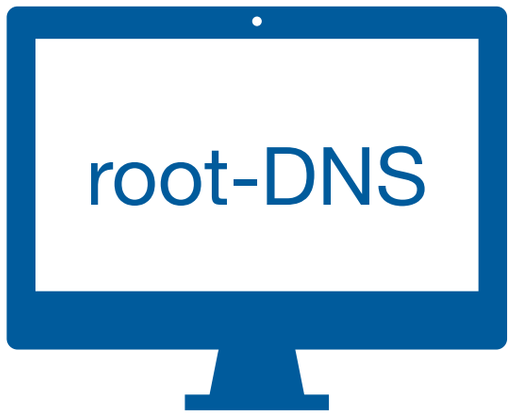
- host.subdomain.example.org ⇢ FQDN
  - Hostname
  - Domain
  - Top-Level
- Note: www.example.org ≠ example.org

Concept

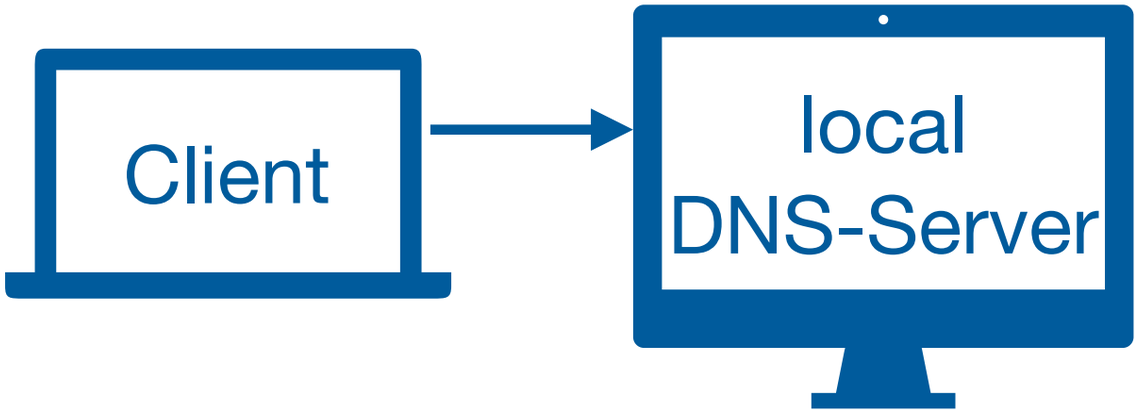
Queries
www.example.org



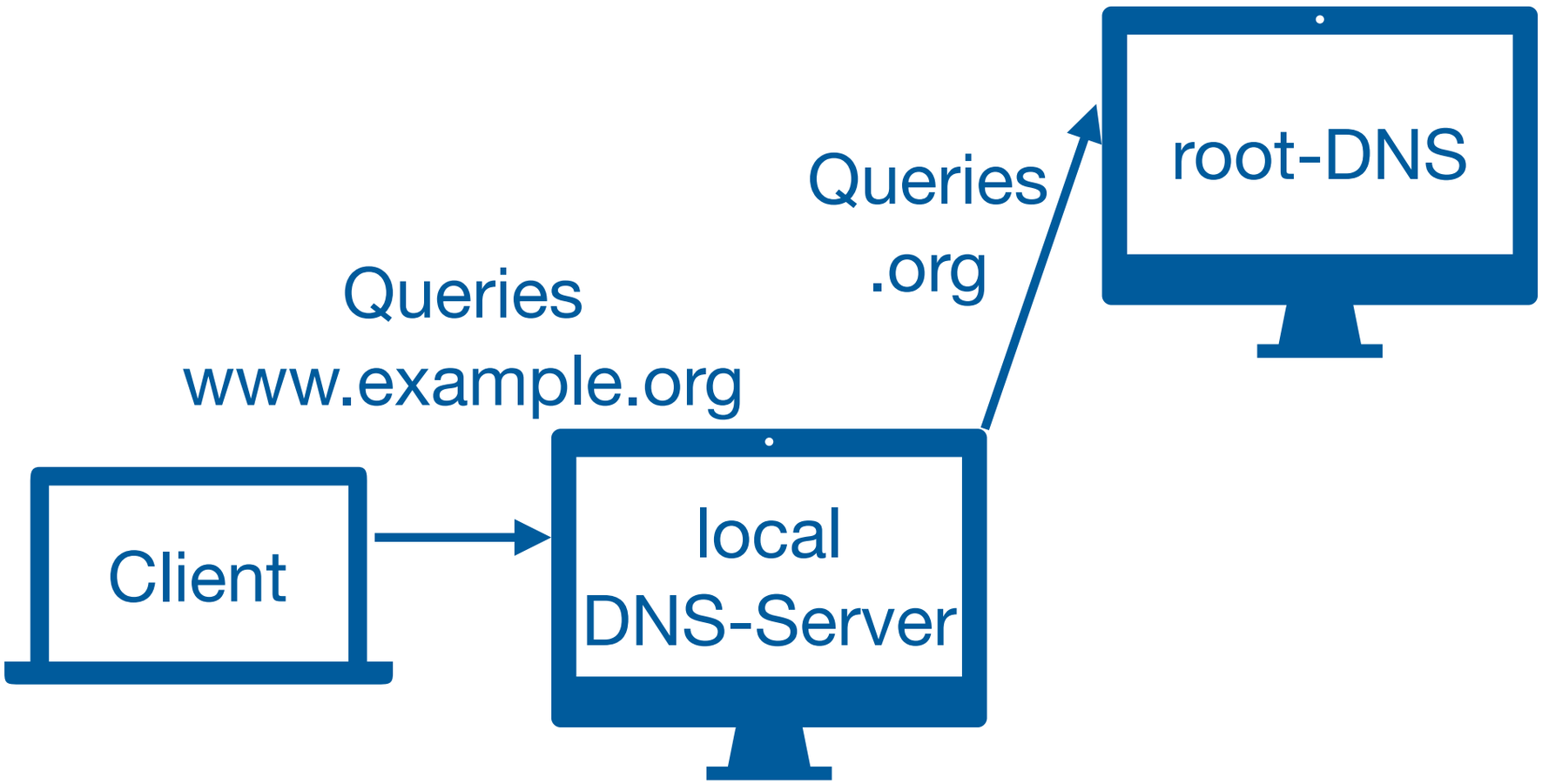
Concept



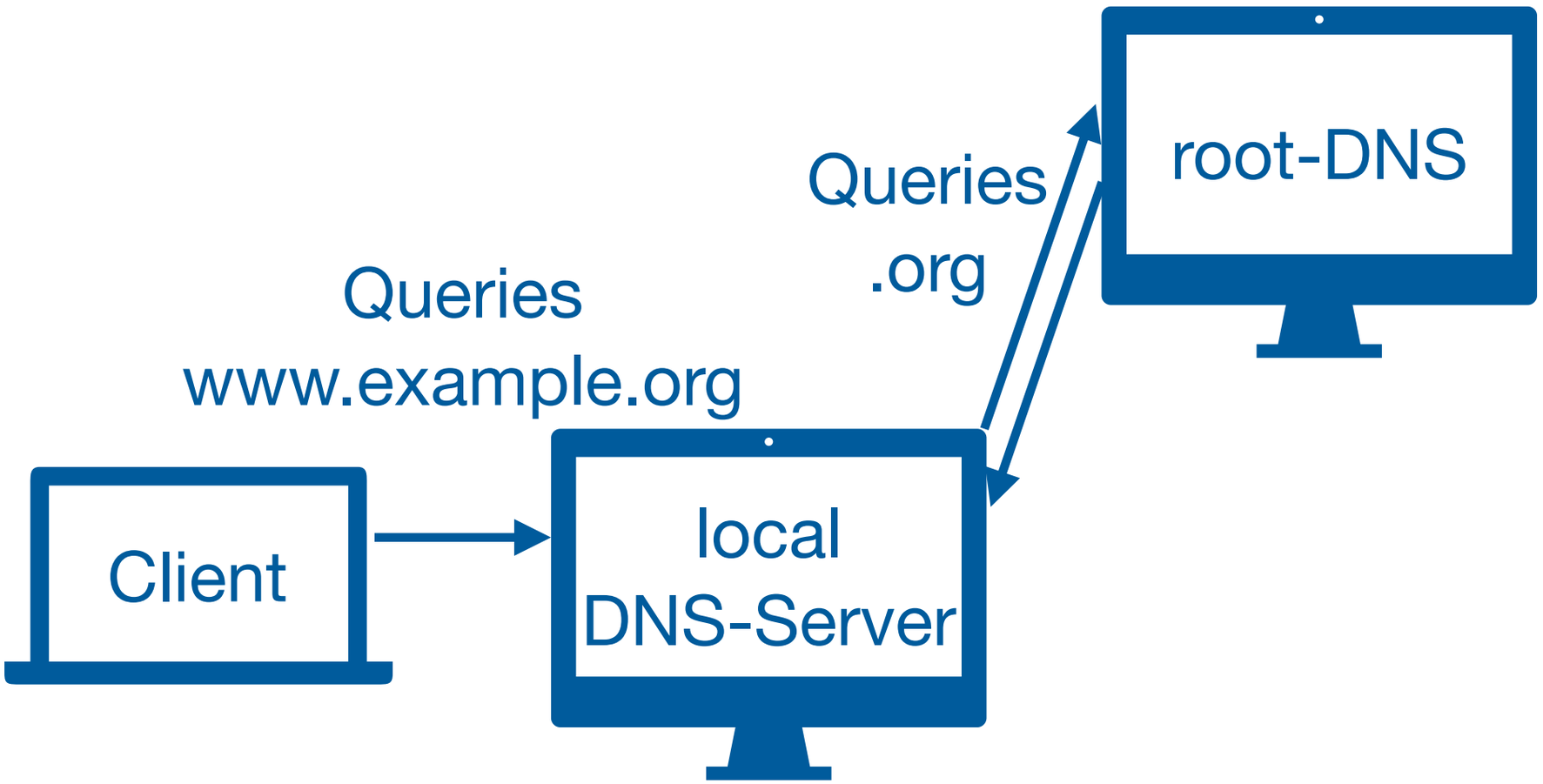
Queries
www.example.org



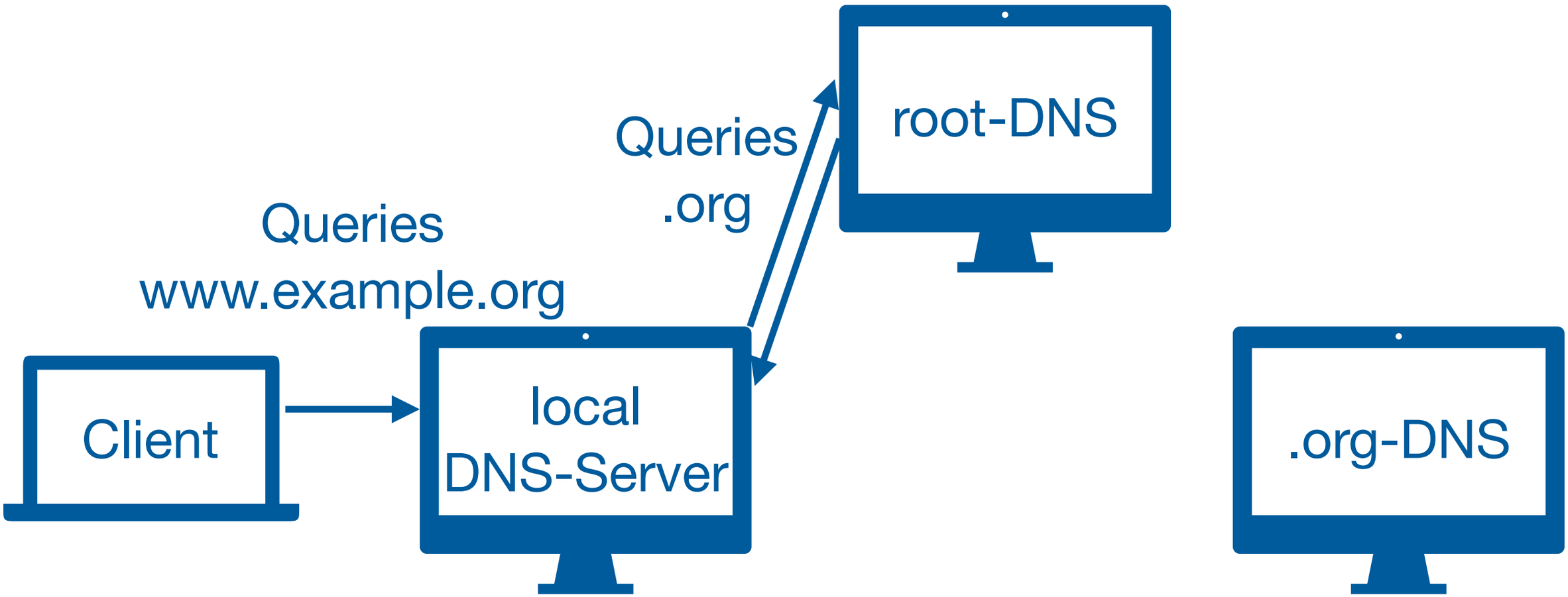
Concept


root-DNS
Queries
.org
Queries
www.example.org
Client
local
DNS-Server

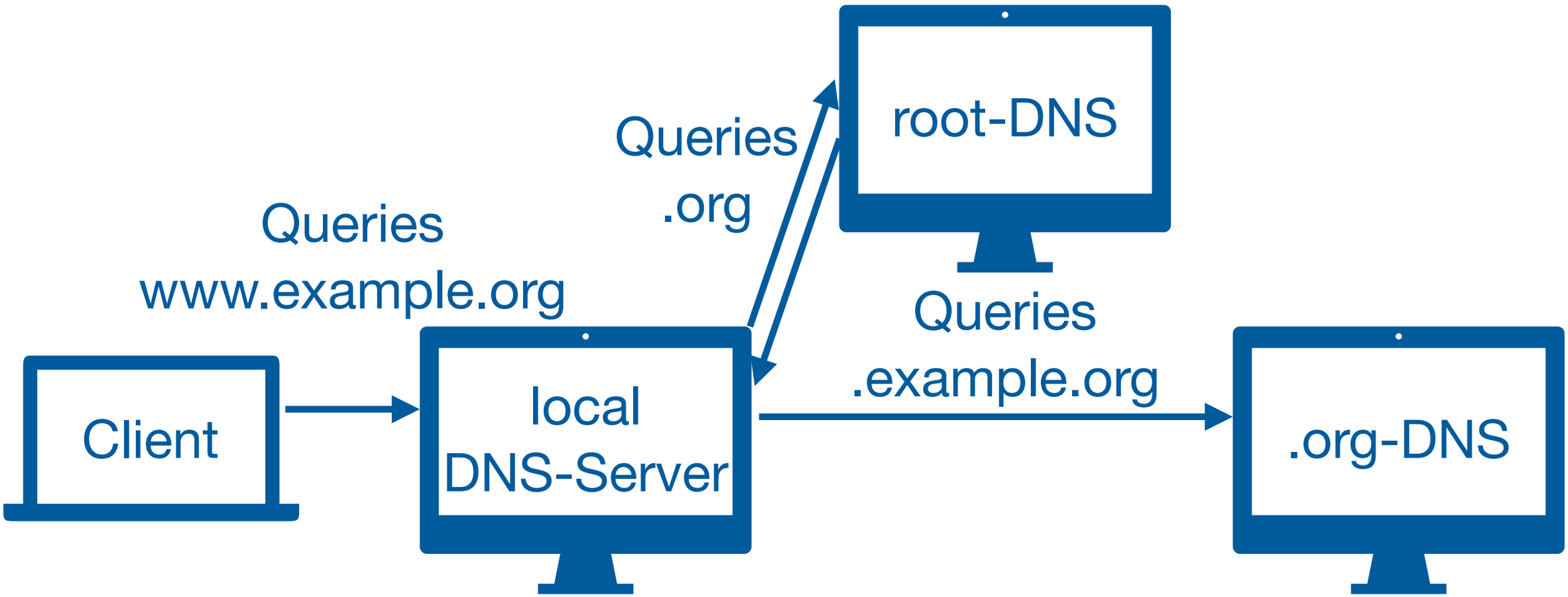
Concept


root-DNS
Queries
.org
Queries
www.example.org
Client
local
DNS-Server

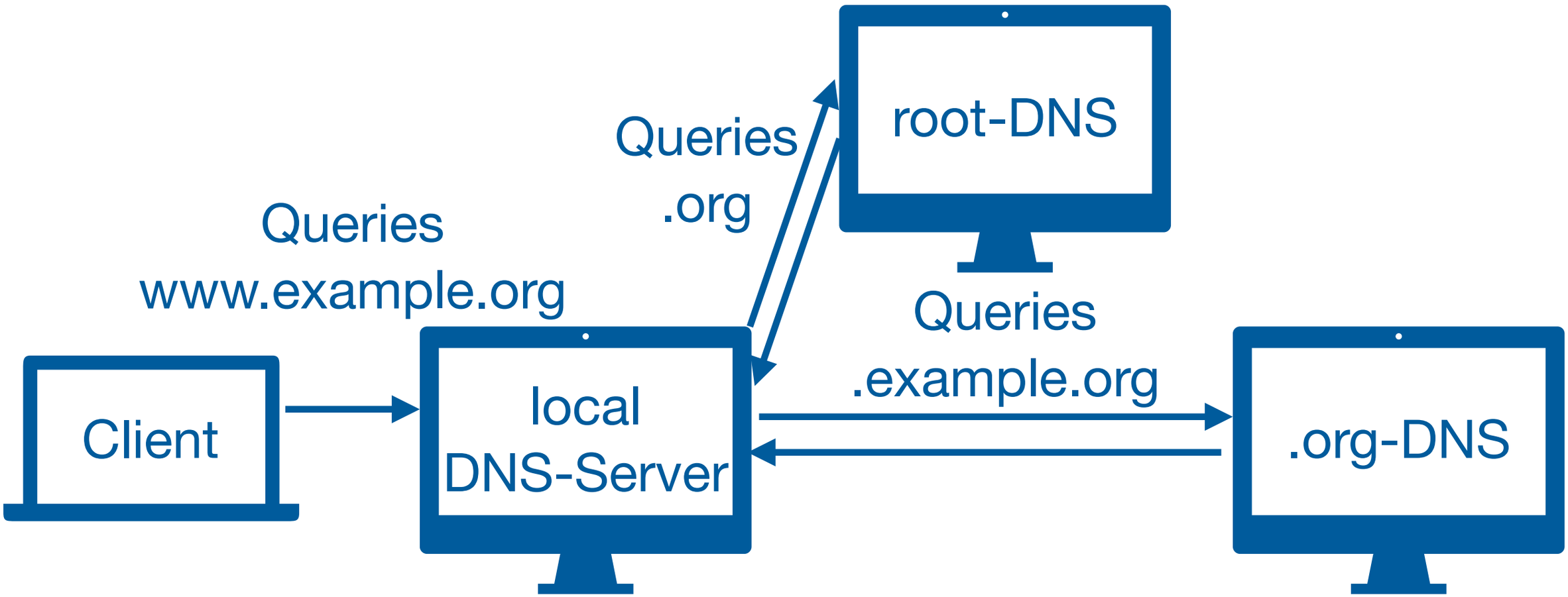
Concept


root-DNS
Queries
.org
Queries
www.example.org
Client
local
DNS-Server
.org-DNS

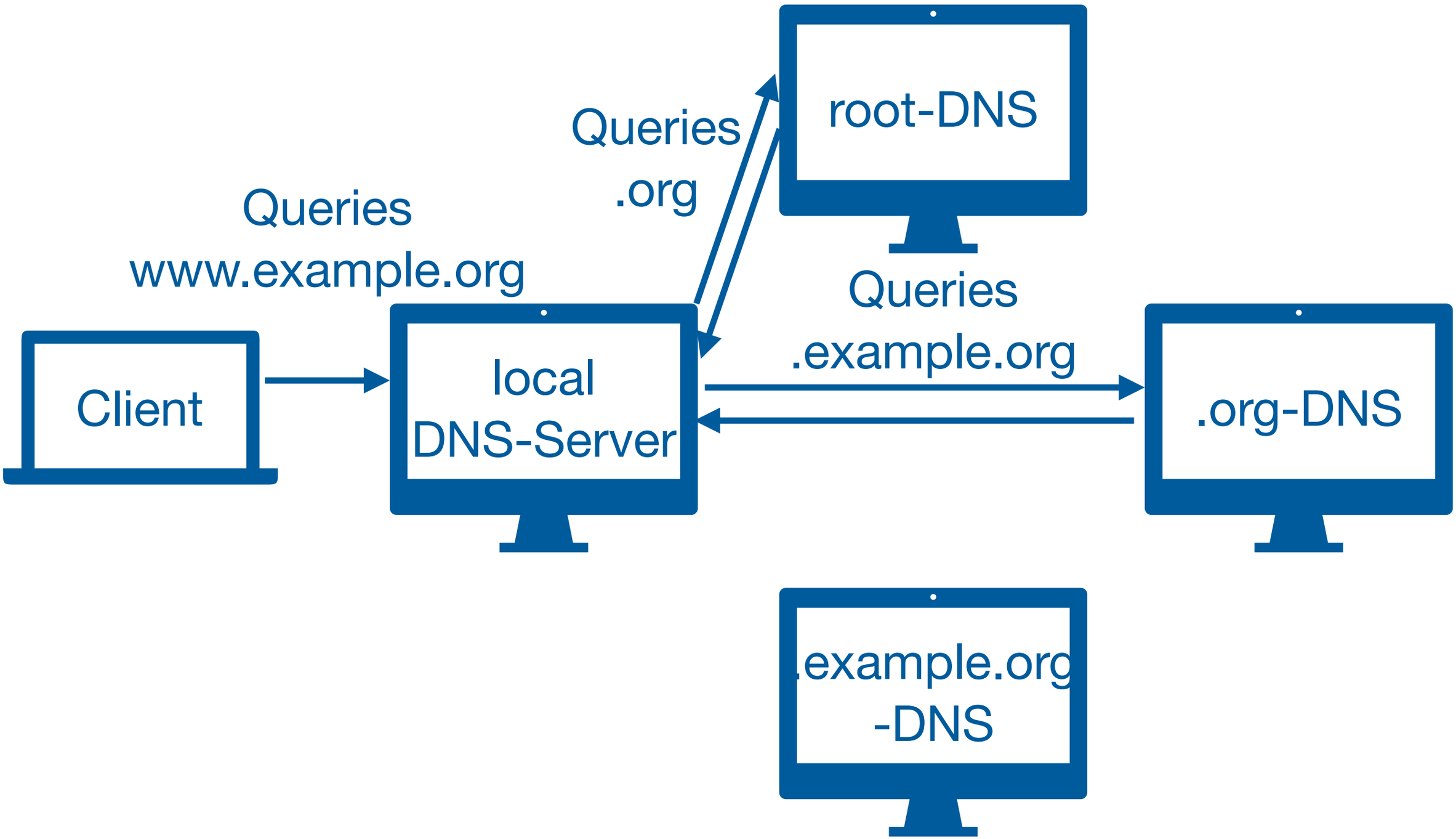
Concept


root-DNS
Queries
.org
Queries
www.example.org
Queries
.example.org
Client
local
DNS-Server
.org-DNS

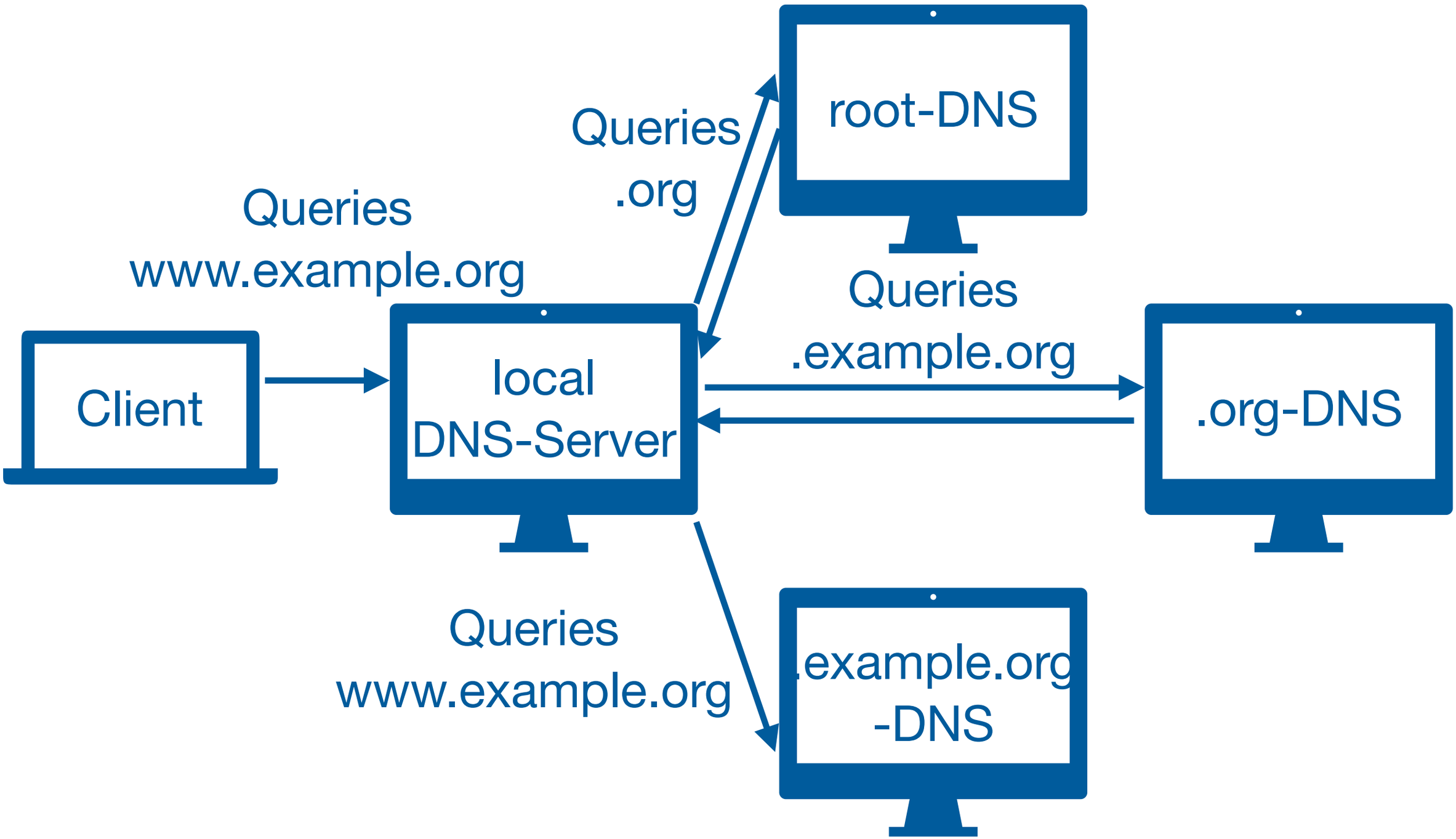
Concept


root-DNS
Queries
.org
Queries
www.example.org
Queries
.example.org
Client
local
DNS-Server
.org-DNS

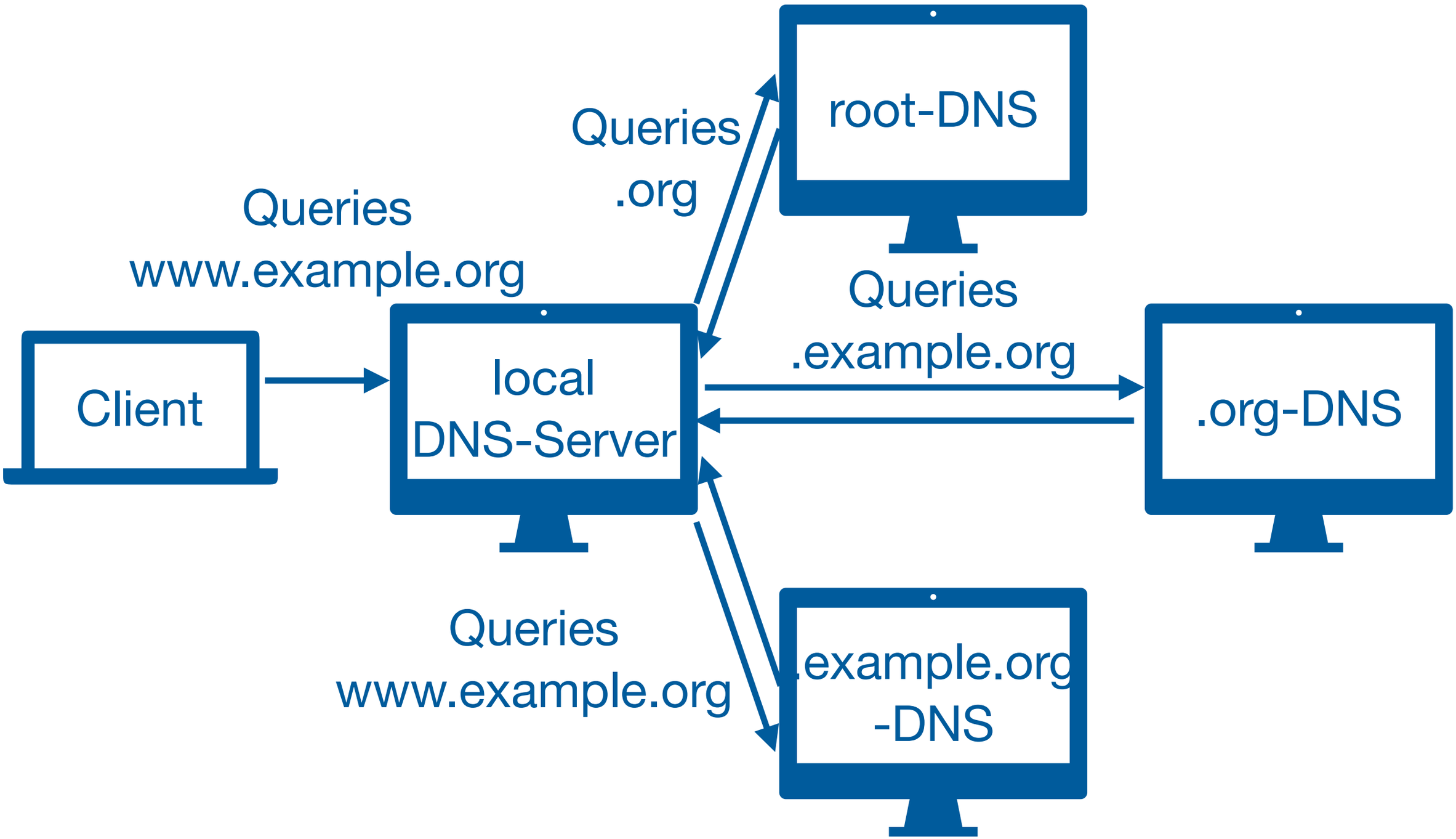
Concept


root-DNS
Queries
.org
Queries
www.example.org
Client
local
DNS-Server
Queries
.example.org
.org-DNS
example.org
-DNS

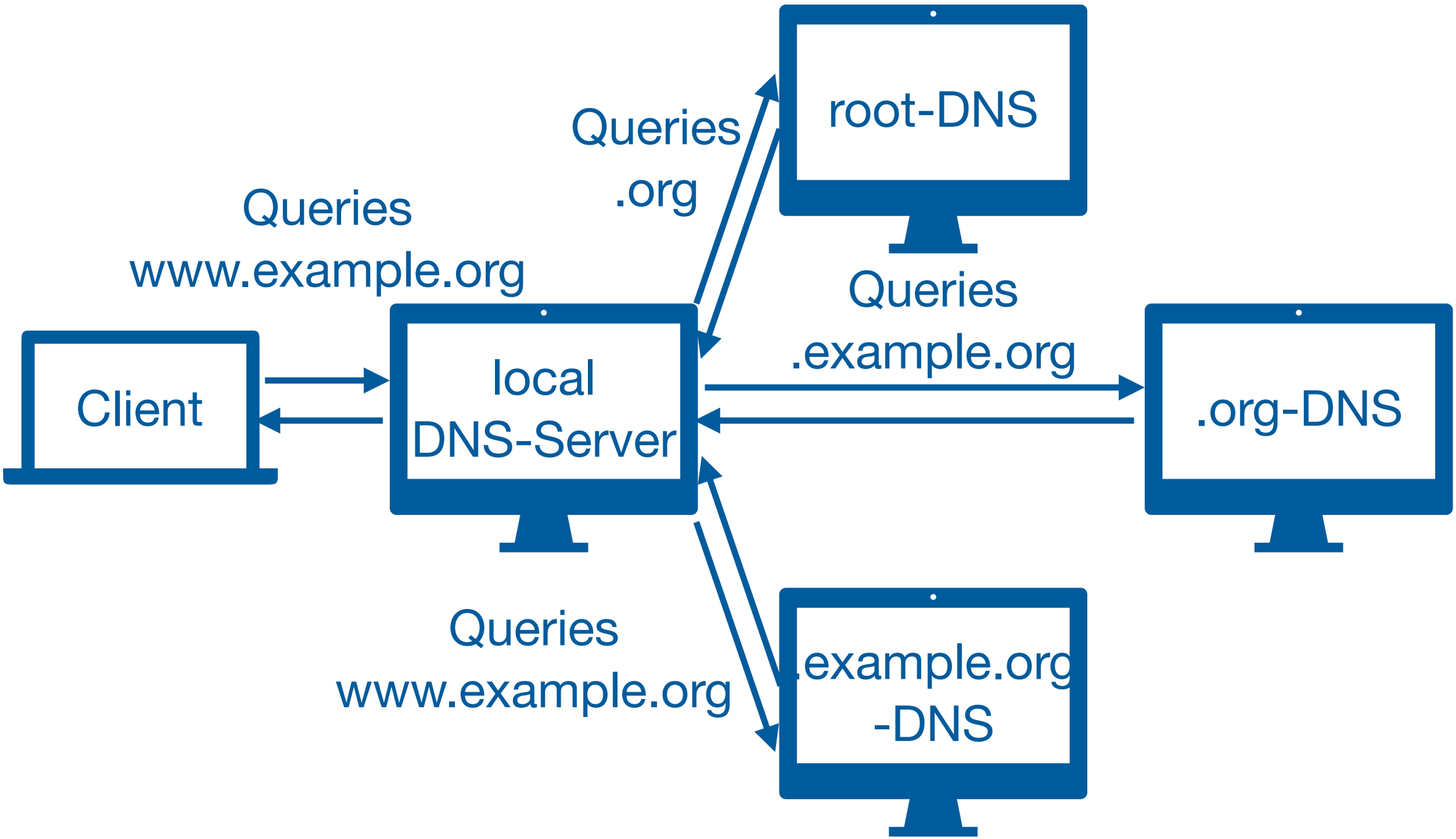
Concept


root-DNS
Queries
.org
Queries
www.example.org
Client
local
DNS-Server
Queries
.example.org
.org-DNS
Queries
www.example.org
example.org
-DNS

Concept


root-DNS
Queries
.org
Queries
www.example.org
Client
local
DNS-Server
Queries
.example.org
.org-DNS
Queries
www.example.org
example.org
-DNS

Concept


root-DNS
Queries
.org
Queries
www.example.org
Client
local
DNS-Server
Queries
.example.org
.org-DNS
Queries
www.example.org
example.org
-DNS

Notes

- Caching, hence
  - TTL (Time To Live)
  - Serial
- Risks
  - DNS Poisoning

Relevant Protocols

- HTTP
  HyperText Transfer Protocol
  - Version 1.0 / 1.1
  - HTTP/2
  - HTTP/3
- HTTPS
  HTTP Secure ➜ HTTP + TLS

Do you speak HTTP 1.1?

- Methods
  - GET
  - POST
  - HEAD
  - PUT / PATCH
  - DELETE
  - TRACE
  - OPTIONS
  - CONNECT

Note: Host-Header

- Name-Based Hosting (since HTTP/1.1)
- prior to 1.1:
  - IP based
  - Port based

Requested Ressource

- Path and filename, e.g.
  - /index.html
  - /some/sub/dir/pic.jpg

GET

- Syntax: GET *Ressource* HTTP/*Version*
- Examples:
  - `GET / HTTP/1.1`
  - `GET /banking/send_money.php?amount=100&dest=123456789`

- GET might send further information as request parameters

GET Request Header

- Content-Type
- Accepted-Language
- User-Agent
- …

POST Request

- Syntax: POST Ressource HTTP/Version
- POST needs Content-Encoding Header
- DATA is sent separately

HEAD-Request

- Same as GET, but will only result in the Header, not the Data
- e.g. to check Proxy-Cache validity

PUT, PATCH & DELETE

- Exotic for „regular“ web pages
- PUT uploads a new file
- PATCH uploads a patch to an existing file
- DELETE deletes a ressource
- Useful in RESTful

TRACE

- Often not allowed
- Sends received request headers back
- Used for debugging

CONNECT

- Used for ⇗Proxies

HTTP-Statuscodes

- 1xx - Information (please hold the line)
- 2xx - Ok
- 3xx - Redirect
- 4xx - Error (Client)
- 5xx - Error (Server)

HTTP Pipelining

- Send multiple request in a single connection
- Responses in this order

HTTP is stateless

- Each request is sent separately
- No connection information

- Added complexity for web-shops, web-chat, social media etc.

Solution: Cookies

- Information stored on the client
- Sent with each request
  - only to requesting server

How and why could cookies „spy“ on you?

HTTP/2

- RFC 7540
- Supports
  - Multiplexing to prevent Head of Line Blocking
  - Header compression
  - Server push
  - supports QUIC

QUIC (aka TCP/2)

- Transport Layer, but based on UDP
- Allows multiplexed connections
- Lower connection overhead
- Better adaption to network changes through ConnectionID
- Enhances performance

HTTP/3

- Based on QUIC
- Same semantics as HTTP/2 and HTTP/1.1
- Usage-Stats: https://w3techs.com/technologies/comparison/ce-http2,ce-http3

General Requirements

WebServer

Exercise - Group Work

- Implement a tiny web server
  - Accepts connections on port 80
    - No SSL
    - Supports only GET on HTTP/1.1
    - Ignores all headers but the Host header
- Any language you like - choose wisely and find good reasons for your choice.

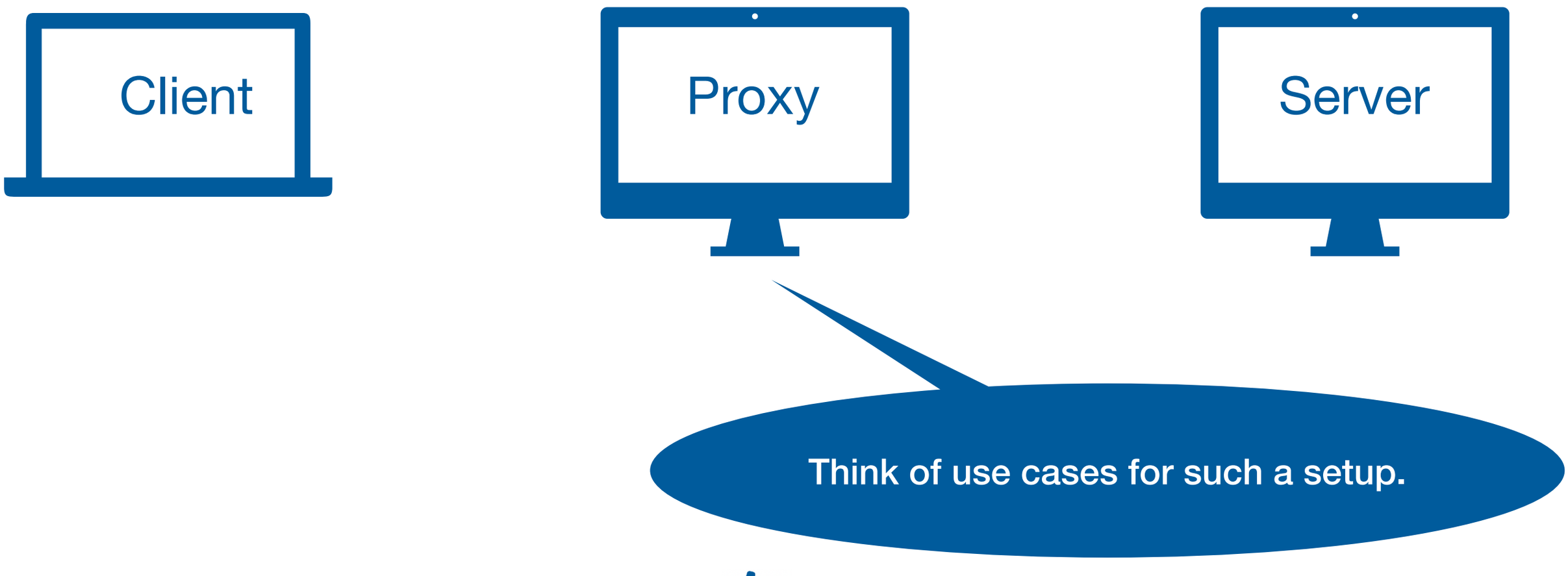
Preparation: What a WebServer should do

- Serve Webpages
  - Accept TCP Connection
  - Parse HTTP Request
  - Locate / Request Ressource
    - Static (File)
    - Dynamic (Output of a process)
      - Provide an interface
  - Add Headers (Content-Type is important)
  - Send Data
  - Close Connection
- Handle parallel requests
- Additional Features
  - Pipelining
  - TLS
  - Logging
- Security
  - (d)DoS
  - Access Control

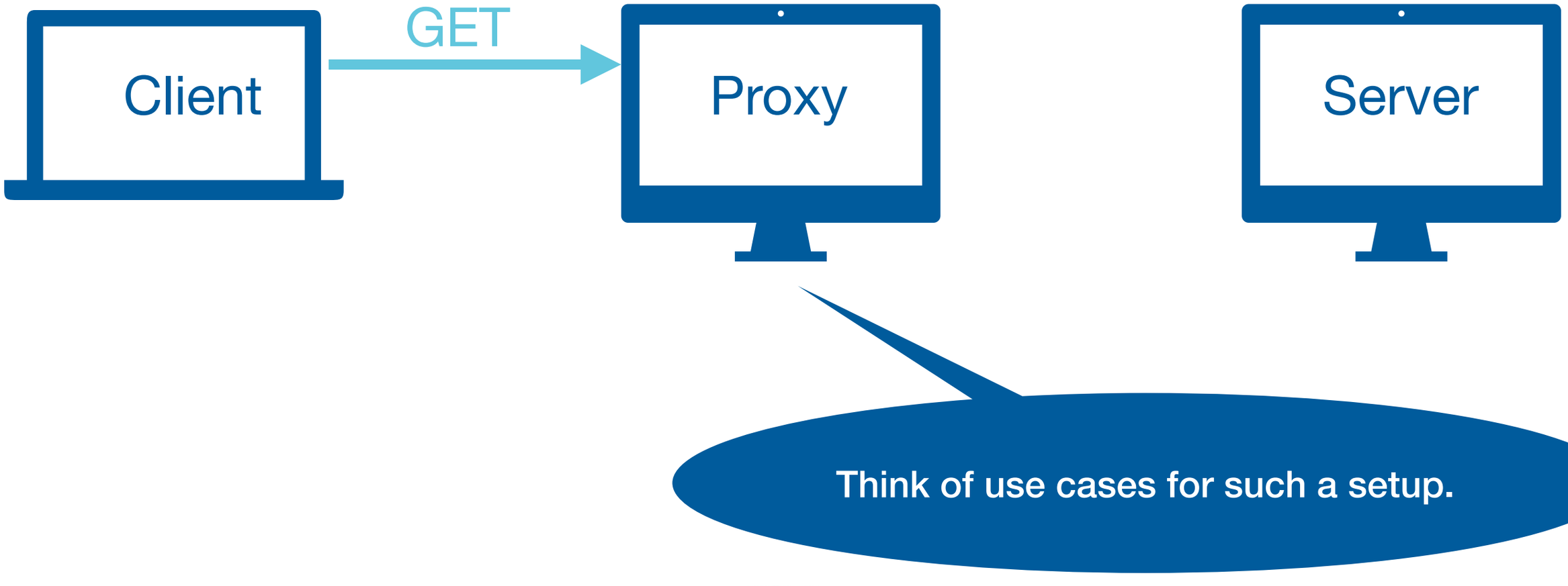
Proxies

- Forward Proxy
- Reverse Proxy

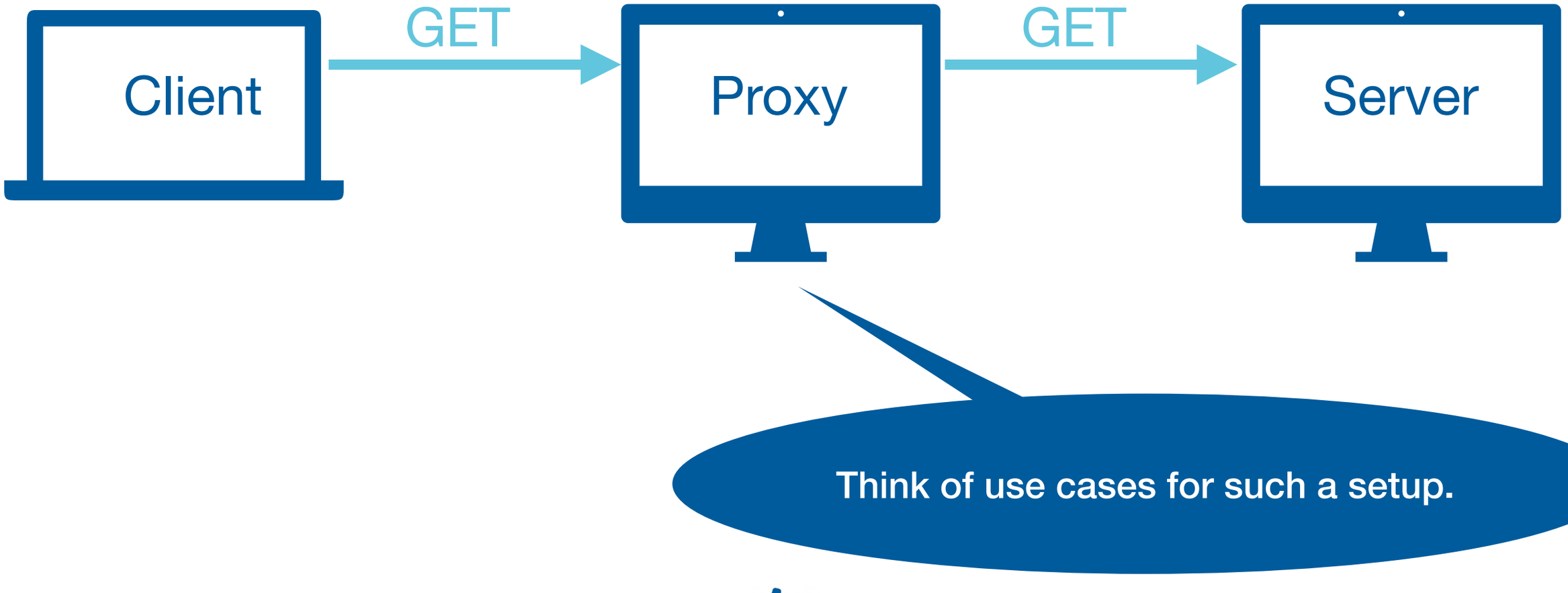
General: Proxy



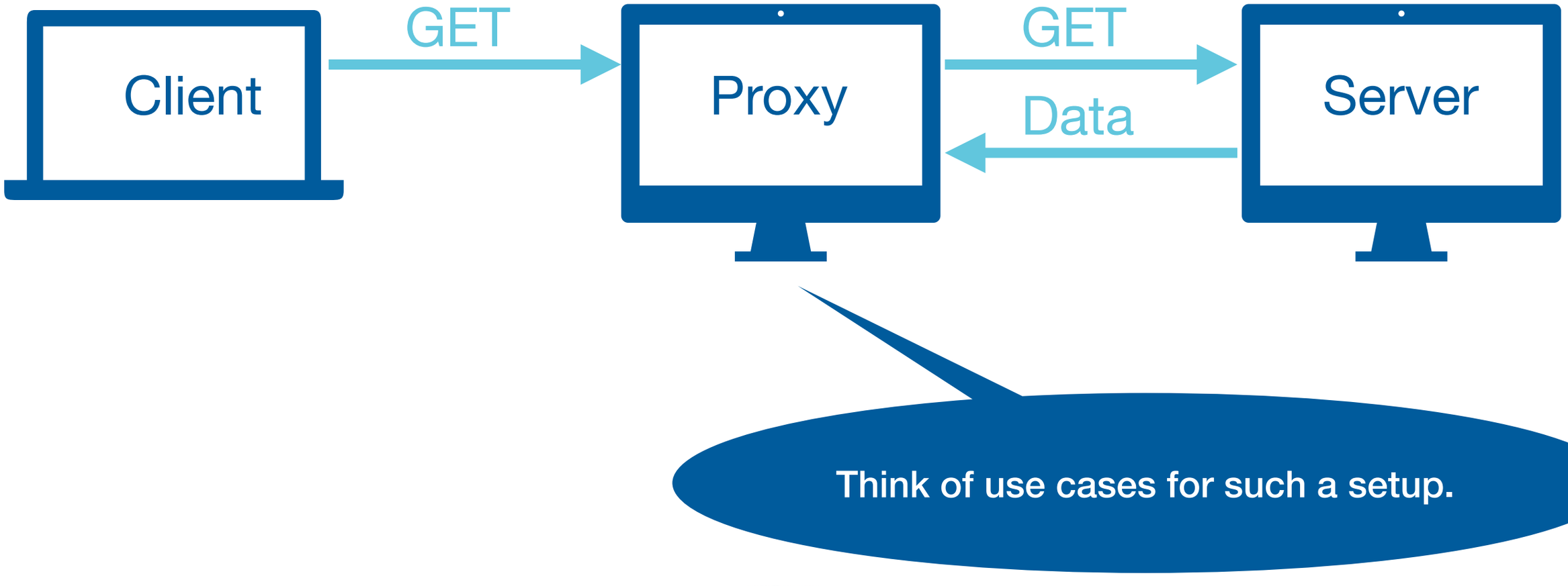
General: Proxy



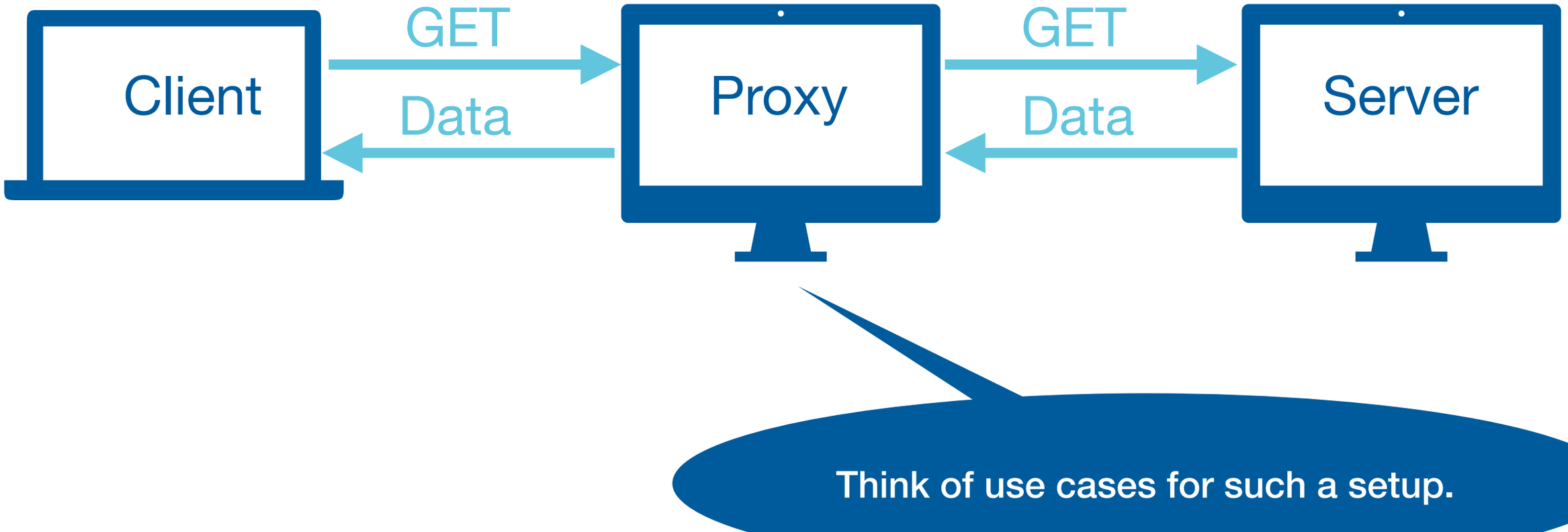
General: Proxy



General: Proxy



General: Proxy

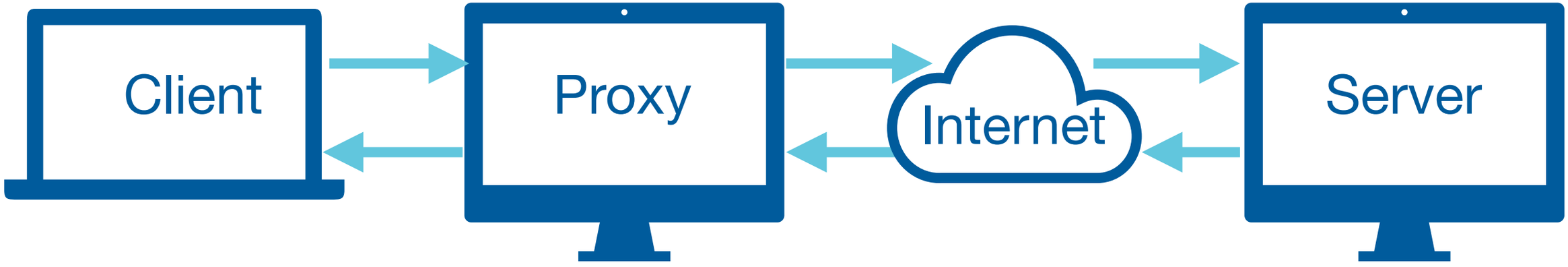


Group work:

- Find use cases for proxies

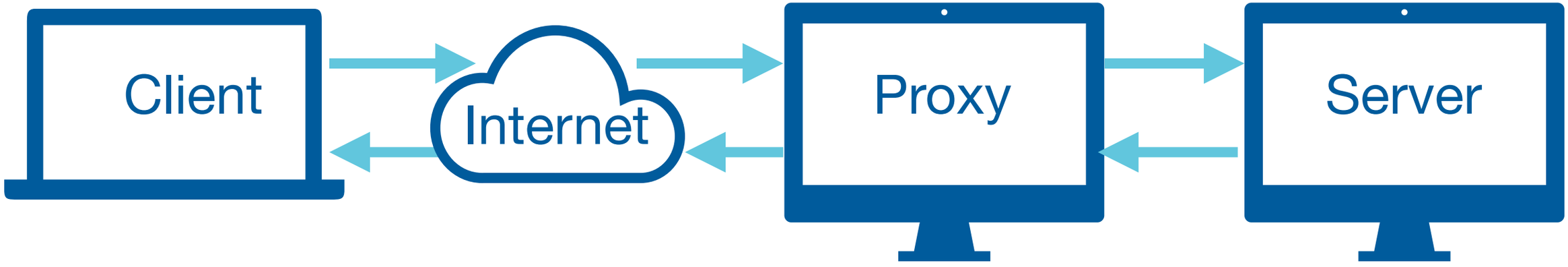
Forward Proxy

Forward Proxy



Reverse Proxy

Reverse Proxy



Group work

- Implement a „phishing proxy“
- Store any credentials sent from the client
- Use any language you deem appropriate (explain why!)

HTML and friends

- SGML
- HTML
- XML
- XHTML
- HTML/5
- CSS
- JavaScript
- DOM

SGML

- Standard Generalized Markup Language
- Meta-Language describing Markup Languages
- ISO 8879
  1999-11: ISO 8879 Technical Corrigendum 2
- EN 28879, DIN EN 28879

SGML

- Tags <example>data</example>
  - with / without Attributes
    - with / without content
- Entities (< " …)
- DTD: Document Type Definition
- and some more

HTML

- SGML derived
- 13.03.1989 first proposed by Sir Tim Berners Lee
  - only supported text
  - Updated to support bold, italics and images
- 1995 HTML 2.0
  - Forms
- 1997 HTML 3.2
  - Text Flow around images
  - Applets
  - Tables

HTML 4.01

- HTML 4.0 18.12.1997
- HTML 4.01 24.12.1999
- Stylesheets
- Frames
- Scripts

CSS

- Cascading Style Sheets
- Separates Content from Layout
- Allows for adaption to different output media

Quick Intro HTML / CSS

Group-Work

- Manually create a simple web page in HTML4.01 that passes the W3C-Check with CSS
- https://validator.w3.org/#validate_by_upload

Prof. Dr. Tobias Eggendorfer

Presentation Group Work

XML

- eXtensible Markup Language
- „cleaner“ syntax than HTML
- Requires a DTD or Schema
- Transformable using XSL
- Addressing of elements with XPath

Simple XML-Document

```
<addressbook>
  <address>
    <name>Jon</name>
    <lastname>Doe</lastname>
    <phone type=„mobile“>+49 123 1234567</phone>
    <phone type=„work“>+49 30 1234567</phone>
    <location type=„work“>
      <street>Sampleway 14</street>
      <poc>12345</poc>
      <city>Sampletown</city>
    </location>
  </address>
</addressbook>
```

DTD - Group work

- Provide a DTD for the example file

Group Work: XSLT

- Provide an XSL to represent the example file in HTML4.01

XHTML

- XML compliant HTML

Group Work

- Provide the example web page programmes in HTML 4.01 as XHTML document
- Validate it with W3C-Validator

Presentation Group Work

HTML 5

- Most recent

JavaScript

- Originally: Simple Client Side Script Language
- Now: Both Client and Server side
- Turing Complete
- Provides features such as Window-Size, Screen Resolution etc.

JavaScript

- Variables / functions must not use reserved names
- Variable != variable (case sensitive)
- Variables do not need to be declared (except in strict mode)

Activating Strict Mode

- 'use strict';

- Note: Strict might be used locally in function only

Comparisons

- ==, =!, <=, >=, >, <
- Special effect:
  - 42 == „42“ (auto type conversion)
  - But: 42 === „42“, 42 !== „42“

Function

- function name (param1, param2) { …; return .. }

Data Structures

- Array
  var names = [‚John‘, ‚Jack‘, ‚Jim‘];
  names.length
-

Calculation

- +, -, *, /
- ** Exponent (a**n ↔ a^n)
- % Modulo (a % n ↔ a mod n)
- +=, -=, *=, … (a += b ↔ a = a + b)
- ++, — (Post- and Pre-In-/Decrement)

Nullish coalescing operator ??

```
let a = null  ?? 'default value'
let b = false ?? 'default value'
```

```
console.log(a) // default value
console.log(b) // false
```

ternary operator

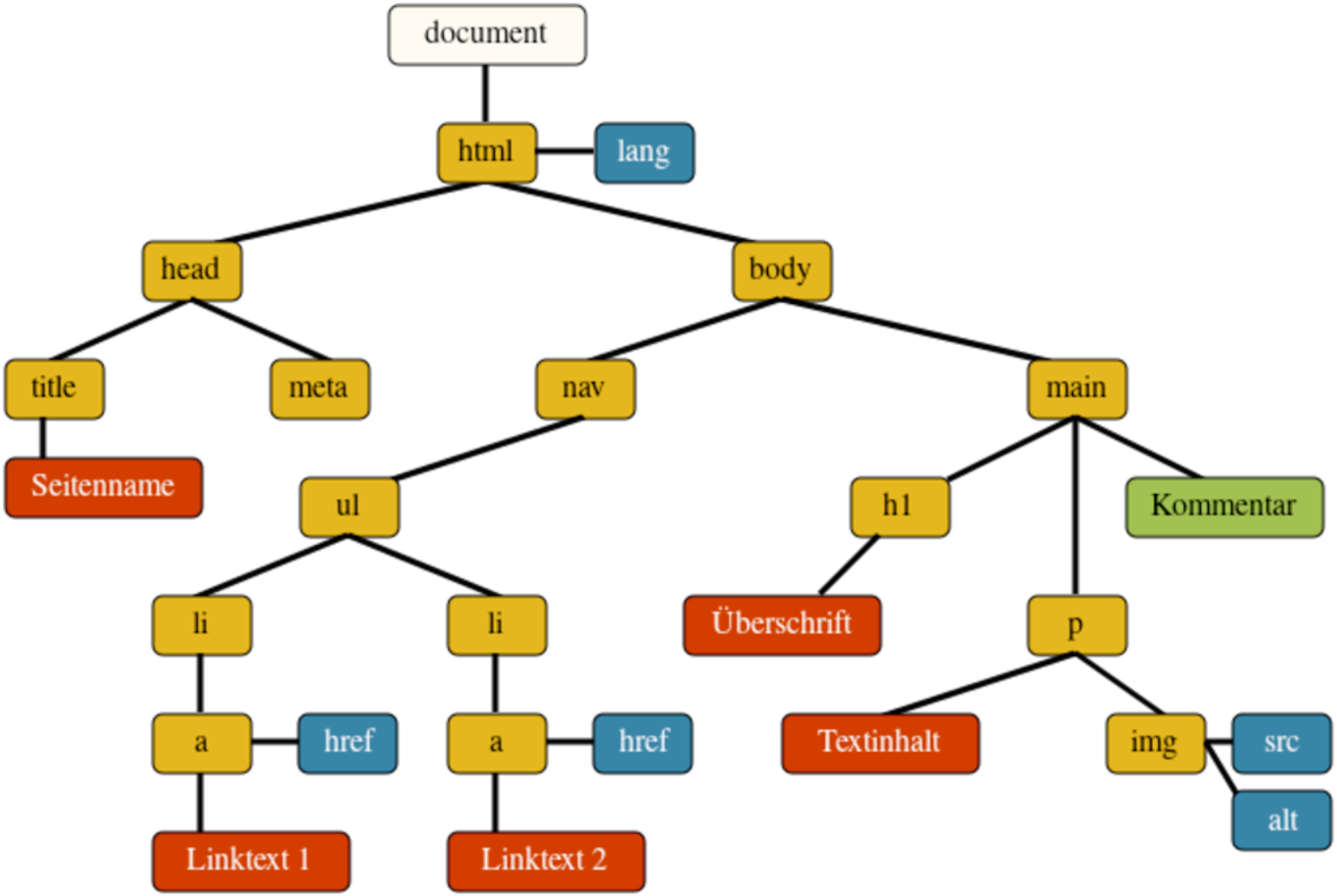
```
a = (x==y) ? x : y
```

```
if (x == y) then
  a = x
else
  a = y
```

DOM

- JavaScript Document Object Model addresses each element of a website in a tree like structure
- Allows for read / modification / addition

Example


document
html
lang
head
body
title
meta
nav
main
Seitenname
ul
h1
Kommentar
li
li
Überschrift
p
a
href
a
href
Textinhalt
img
src
alt
Linktext 1
Linktext 2

Searching

- getElementById()
- getElementByName()
- getElementByTagName() (e.g. find 17th paragraph)
  document.getElementsByTagName(‚p')[17]

Change Tag Content

```
document.getElementsByTagName(‚p‘)[17].firstChild.data =
„This is this paragraph’s new content“;
```

Manipulating

- appendChild
- insertBefore

Group Work

- Implement a simple Calculator using JavaScript in a Webpage
- (Supports +, -, *, /)

Looking at this: What could replace cookies for „spying“?