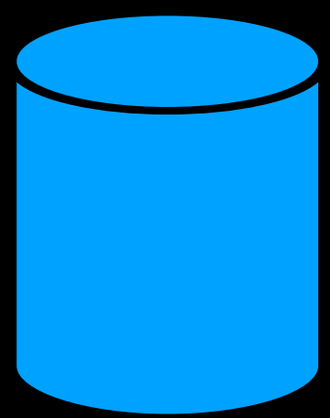
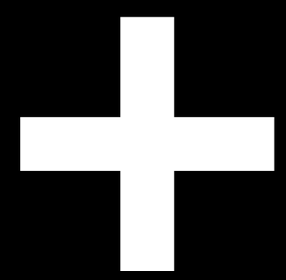
Web scraping

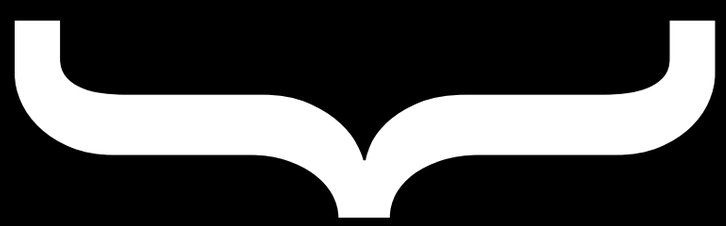
Sharing tech skills

What is web scraping?





Storing web pages
(often also referred to as indexing)



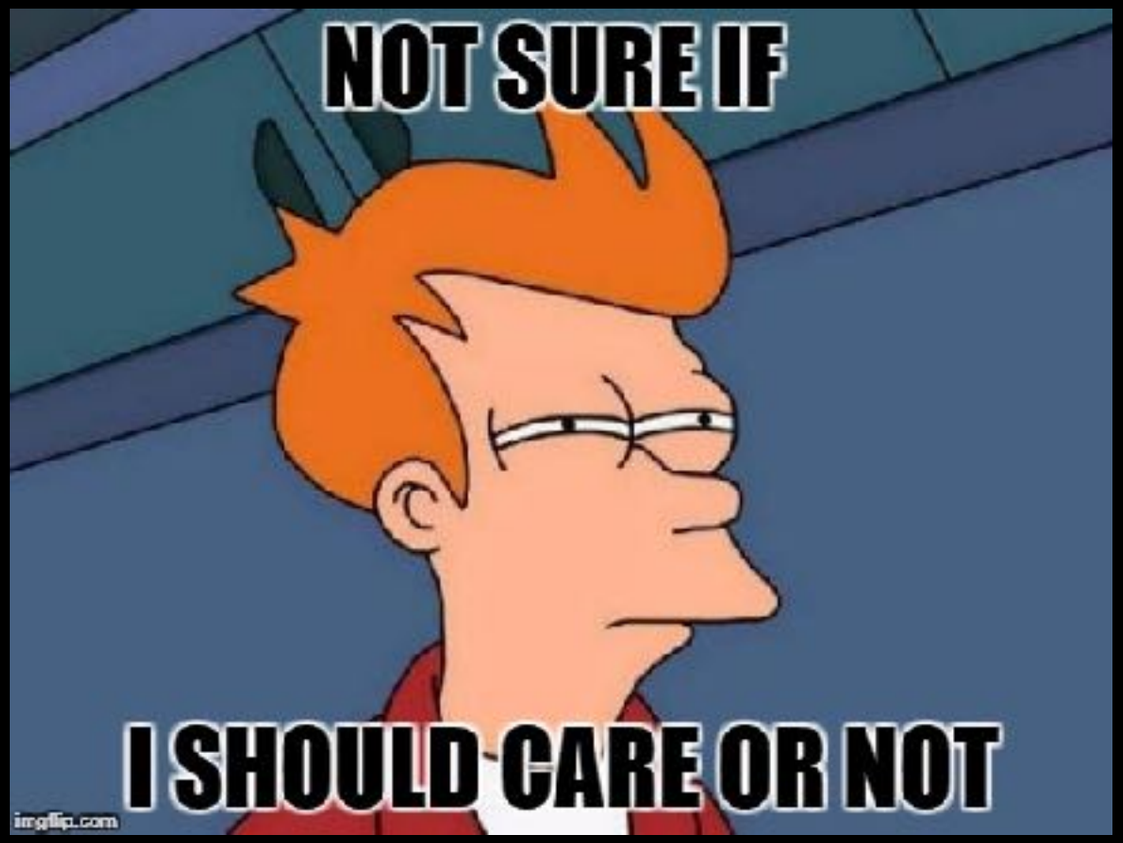
Web crawling

Extracting information from web pages

Source: scientific wild-ass guess

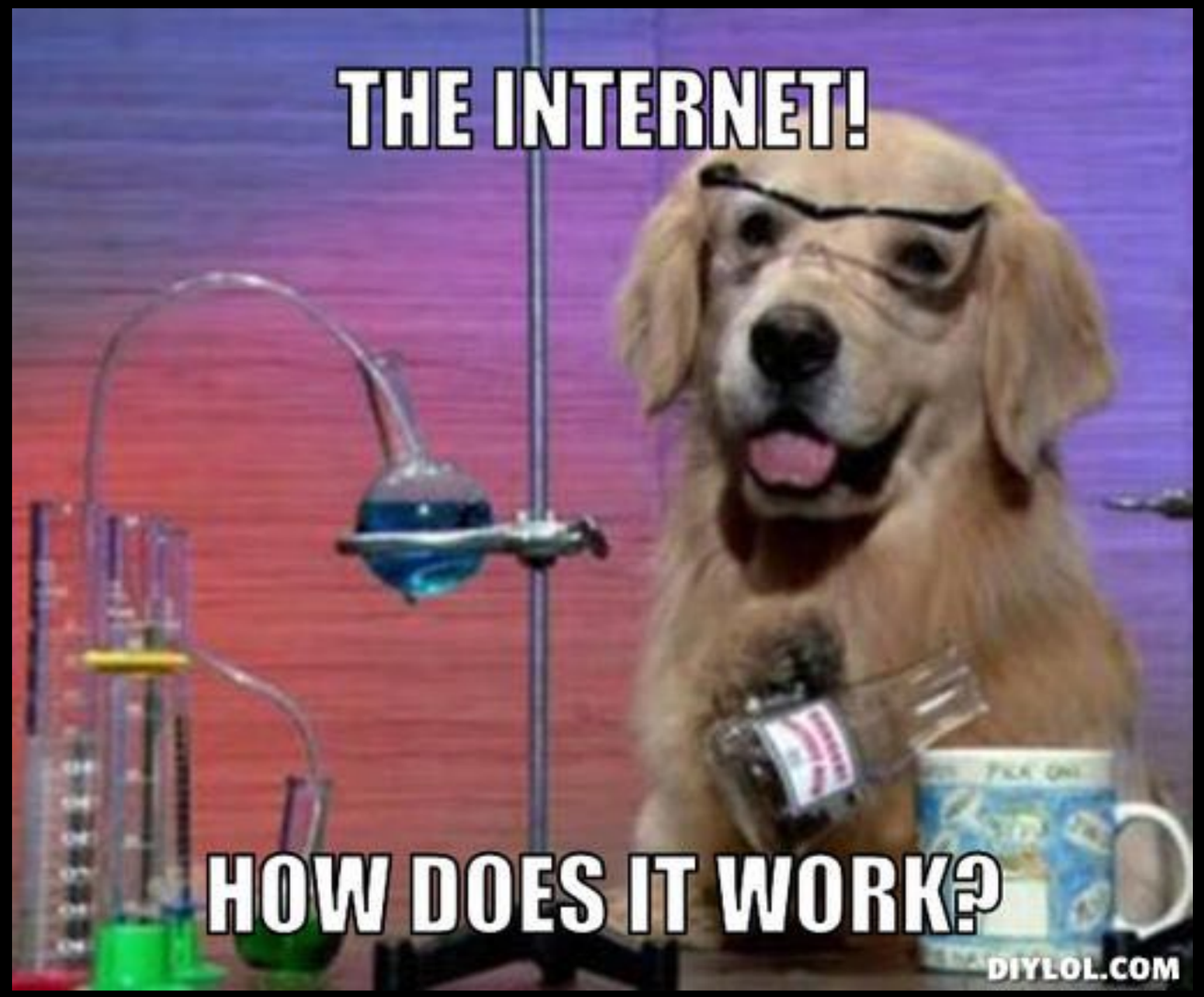
Why should we do web scraping?

- one can build oneself nice little utilities
  e.g. notifications on certain changes in the web, or building a database on a topic
- for data-interested people:
  sometimes it's more exciting to work with data that has not been analyzed by 1000 others before
- one can learn a lot about programming and IT
- it's FUN!!



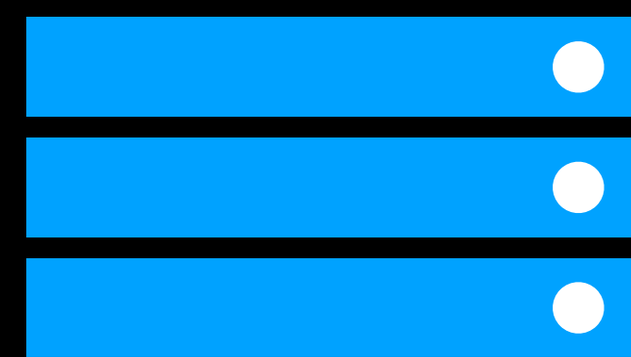
Side note: we'll do web scraping

- It seems like some differentiate data scraping from web scraping: Data scraping may comprise even more data sources, like databases or file systems. We, however, **focus on web pages**!


THE INTERNET!
HOW DOES IT WORK?
DIYLOL.COM

How could one explain the internet better than with many arrows and boxes?

A boxy-arrowed intro to the web



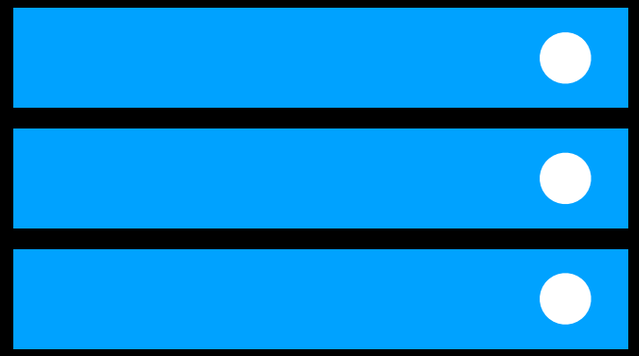
Web server
de.wikipedia.org
(91.198.174.192)

Rosa Mustermann

A boxy-arrowed intro to the web

A boxy-arrowed intro to the web

Domain name server

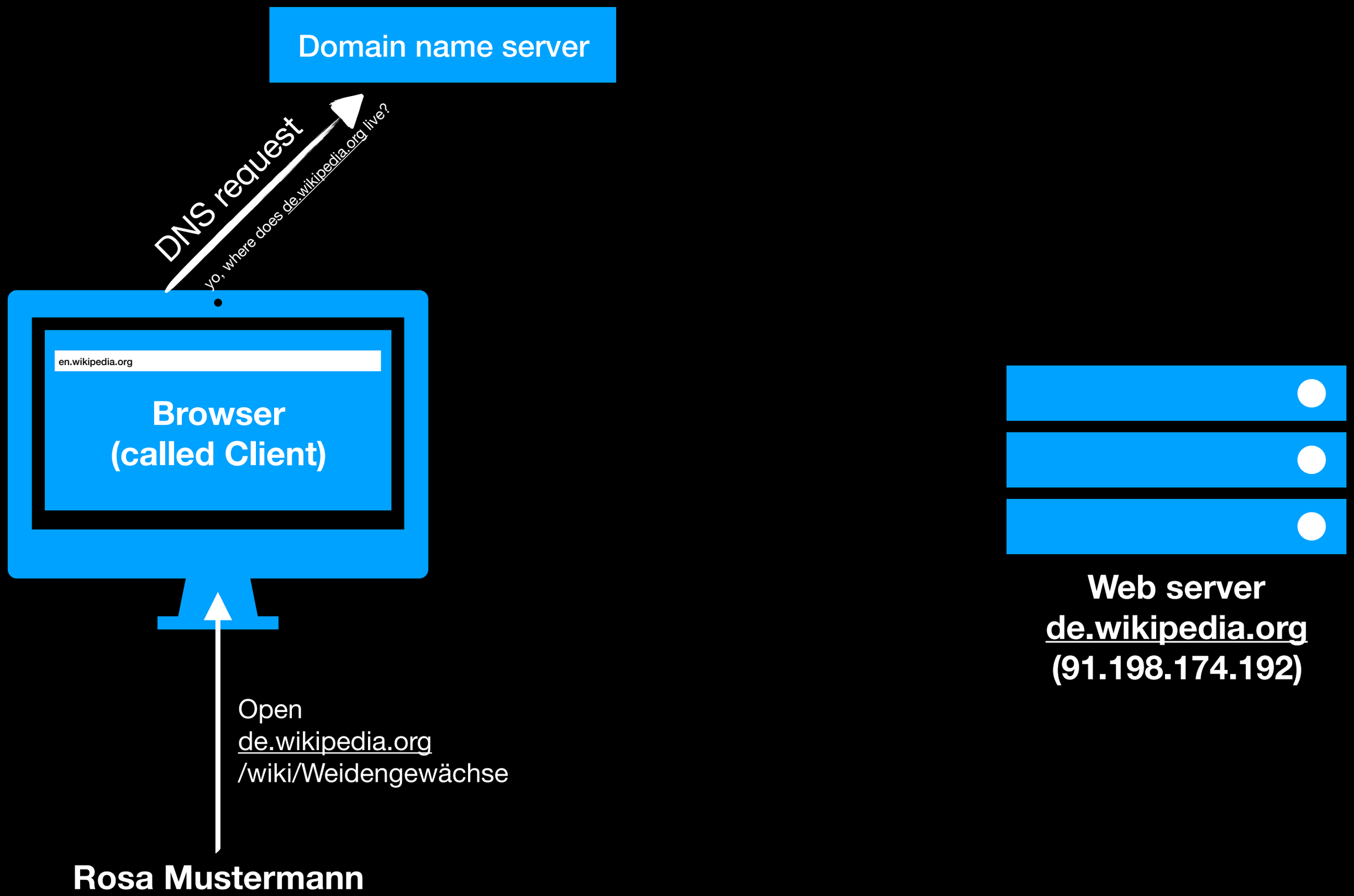


Web server
de.wikipedia.org
(91.198.174.192)

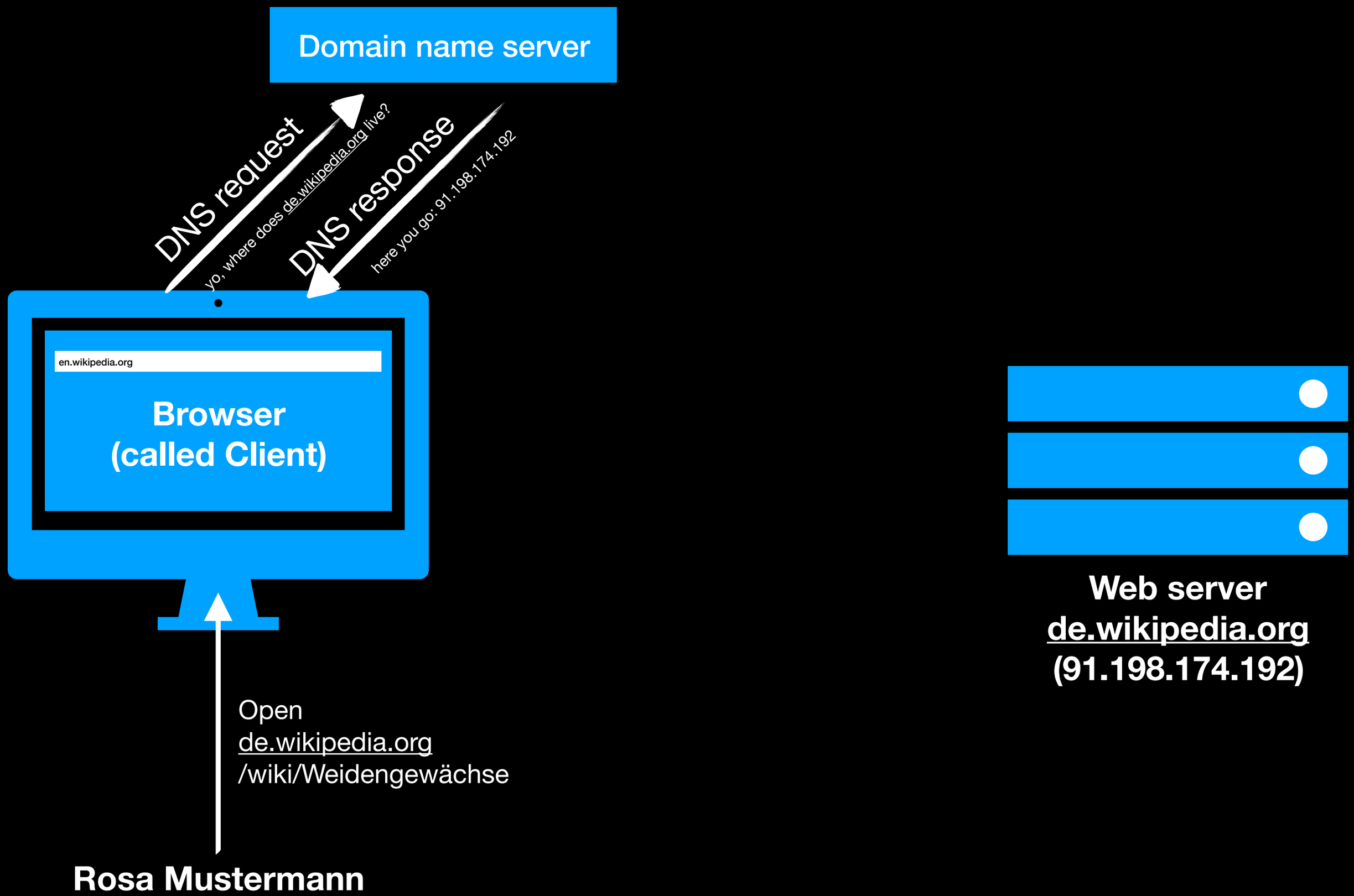
Open
de.wikipedia.org
/wiki/Weidengewächse

Rosa Mustermann

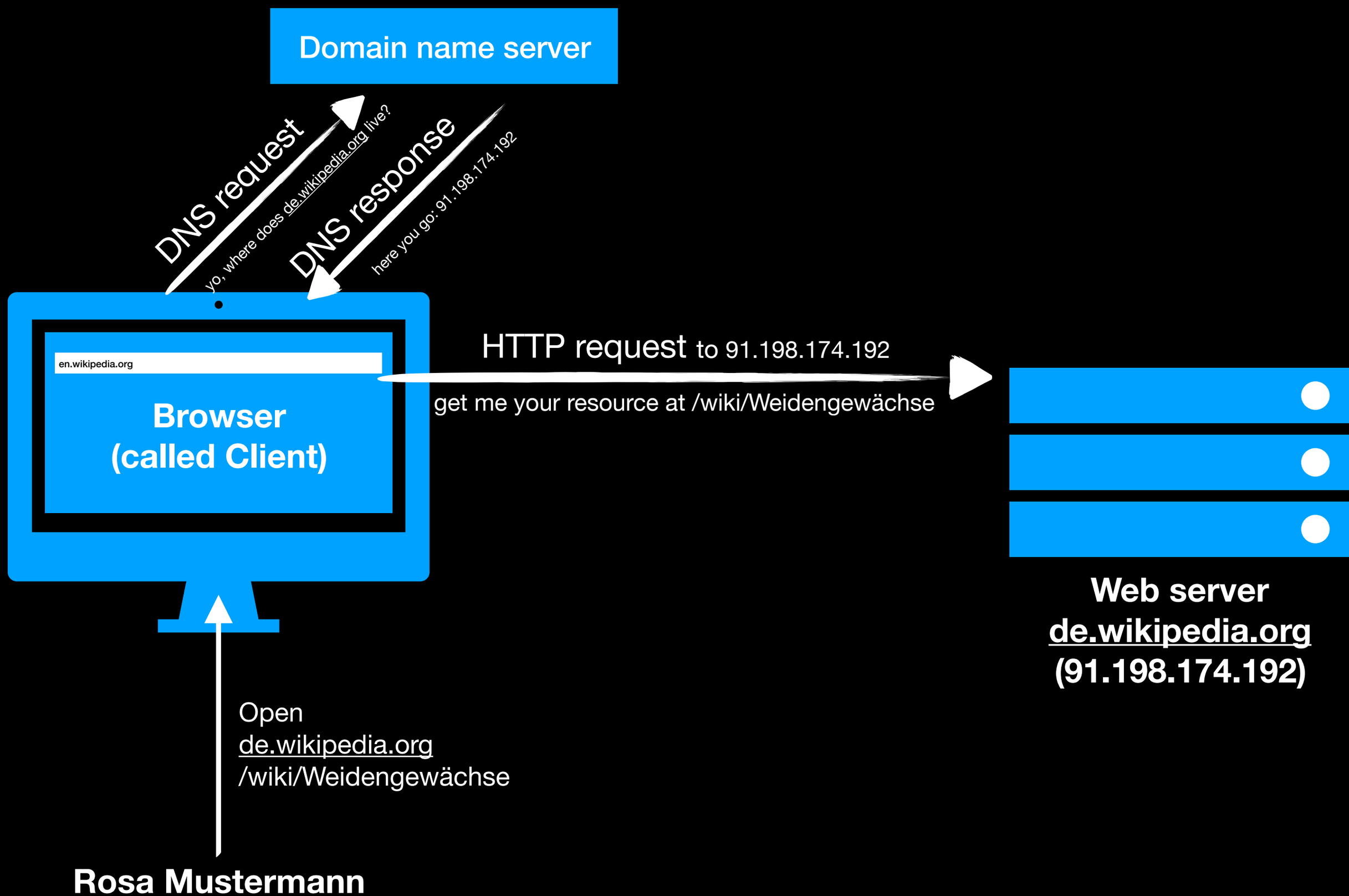
A boxy-arrowed intro to the web



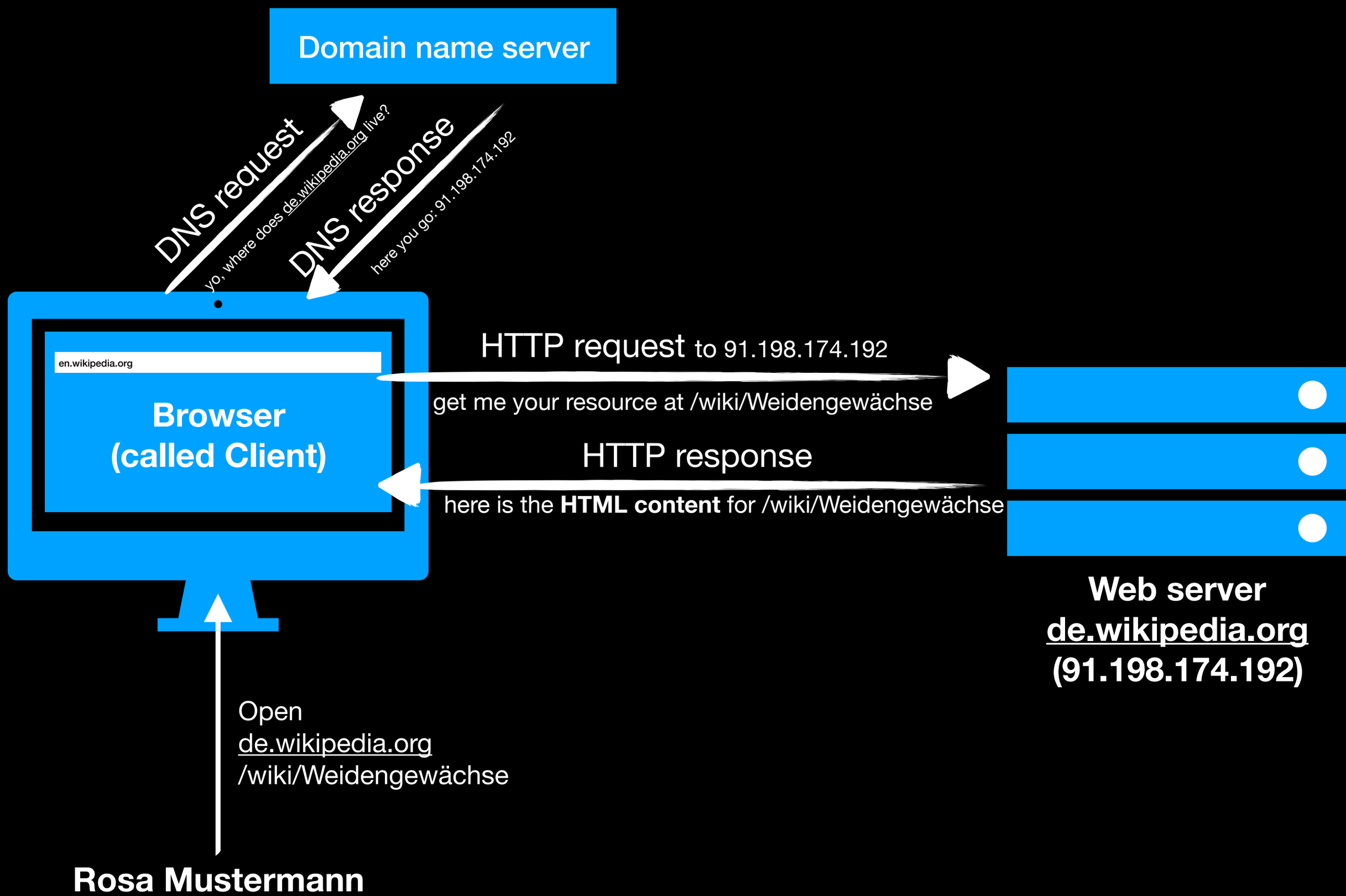
A boxy-arrowed intro to the web



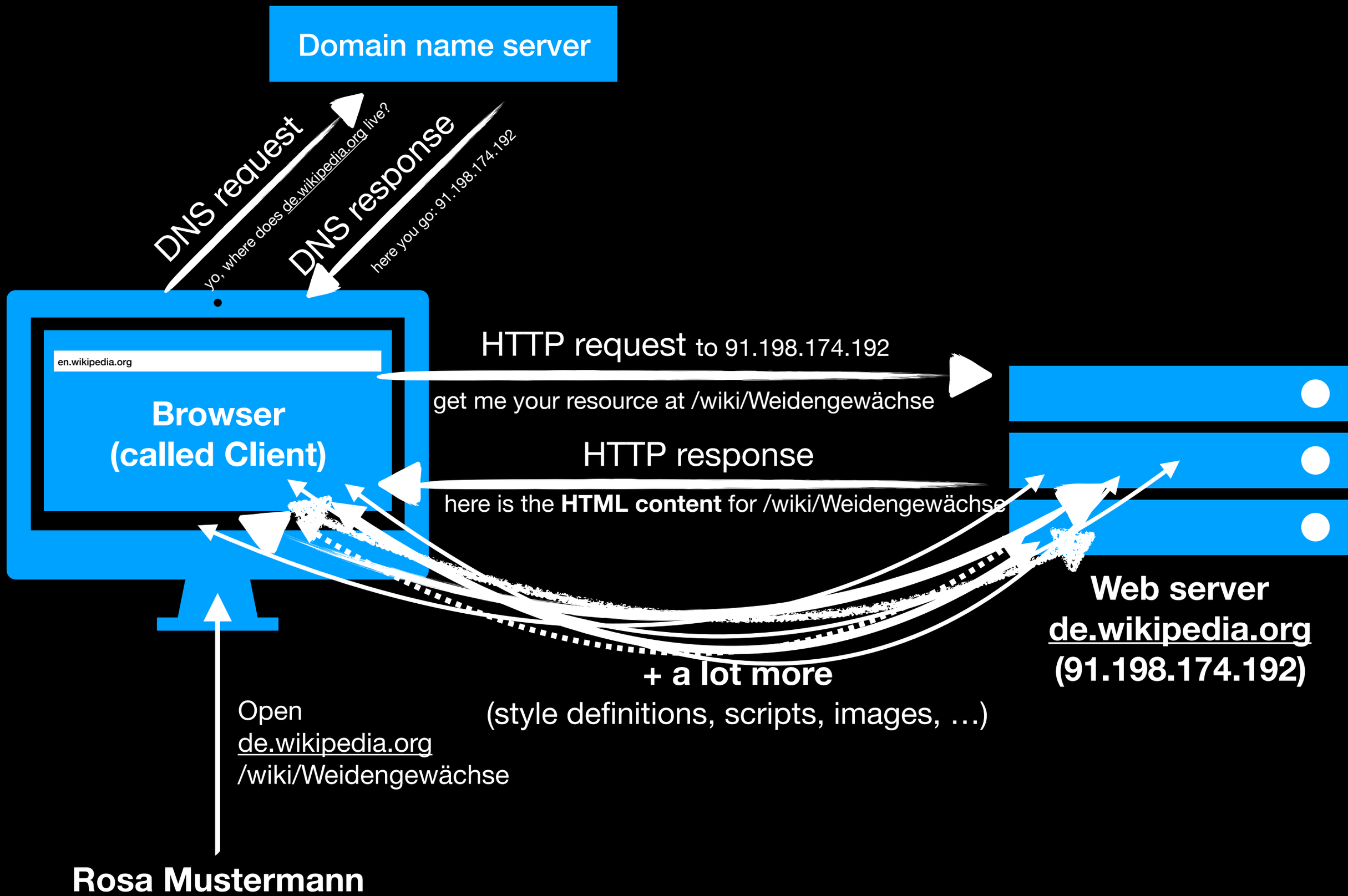
A boxy-arrowed intro to the web



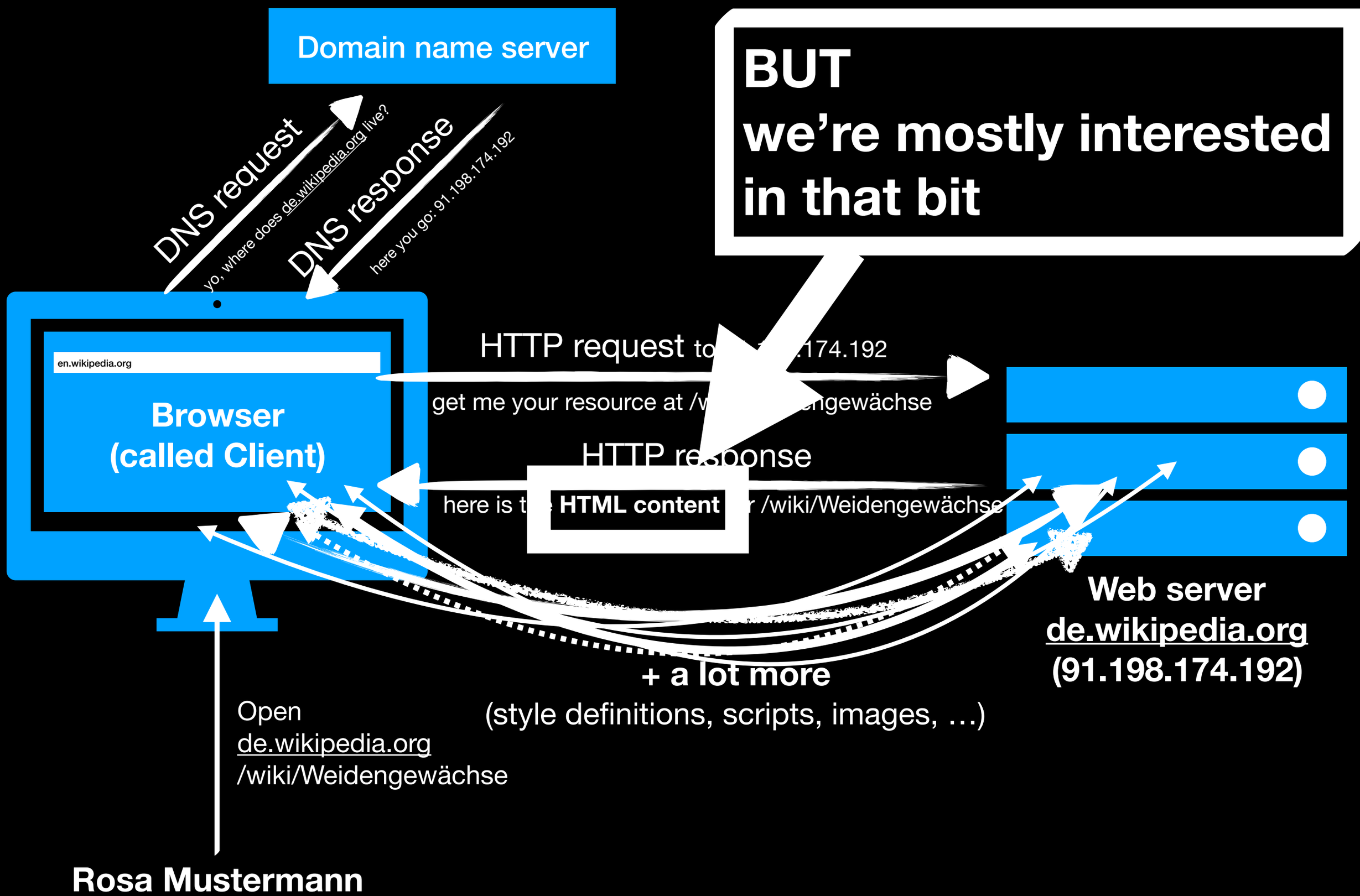
A boxy-arrowed intro to the web



A boxy-arrowed intro to the web



A boxy-arrowed intro to the web



Some definitions

- **HTTP**
  Hyper Text Transfer Protocol
  → Responsible for transferring data between client and server
  → Part of application layer in OSI model (see Appendix A1)
  → Btw: HTTPS is the same but on top of a secure (encrypted) TLS layer

- **HTML**
  Hyper Text Markup Language
  → Language to describe structure and contents of a web page
  → Comparable with LaTeX, Markdown or WikiText

- Enough definitions, ask if something's unclear

Example HTTP <u>request</u>

```
1   GET https://de.wikipedia.org/wiki/Weidengew%C3%A4chse
2   Host: de.wikipedia.org
3   User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:75.0) Gecko/20100101 Firefox/75.0
4   Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5   Accept-Language: de,en-US;q=0.7,en;q=0.3
6   Accept-Encoding: gzip, deflate, br
7   Connection: keep-alive
8   Cookie: WMF-Last-Access=19-Apr-2020; WMF-Last-Access-Global=19-Apr-2020; GeoIP=DE:NI:L__neburg:5
9   Pragma: no-cache
10  Cache-Control: no-cache
```

This is the so-called HTTP request header

Example HTTP <u>response</u>

```
1   HTTP/2 200 OK
2   date: Sun, 19 Apr 2020 14:31:14 GMT
3   content-type: text/html; charset=UTF-8
4   server: mw1351.eqiad.wmnet
5   content-language: de
6   vary: Accept-Encoding,Cookie,Authorization
7   last-modified: Sun, 05 Apr 2020 14:31:14 GMT
8   content-encoding: gzip
9   age: 10487
10  server-timing: cache;desc="hit-front"
11  strict-transport-security: max-age=106384710; includeSubDomains; preload
12  cache-control: private, s-maxage=0, max-age=0, must-revalidate
13  accept-ranges: bytes
14  content-length: 22883
15
16  <!DOCTYPE html>
17  <html class="client-nojs" lang="de" dir="ltr">
18  <head>
19  <meta charset="UTF-8"/>
20  <title>Weidengewächse - Wikipedia</title>
21  <link rel="stylesheet" href="/w/load.php?lang=de&amp;modules=ext.cite.styles%7Cext.flag
22  <script async="" src="/w/load.php?lang=de&amp;modules=startup&amp;only=scripts&amp;raw=
23  <meta name="ResourceLoaderDynamicStyles" content=""/>
24  <link rel="stylesheet" href="/w/load.php?lang=de&amp;modules=site.styles&amp;only=style
25  <meta name="generator" content="MediaWiki 1.35.0-wmf.28"/>
```

So-called HTTP response header

Oh! Look at that neat little HTML down here!

Example HTTP <u>response</u>

```
1   HTTP/2 200 OK
2   date: Sun, 19 Apr 2020 14:31:14 GMT
3   content-type: text/html; charset=UTF-8
4   server: mw1351.eqiad.wmnet
5   content-language: de
6   vary: Accept-Encoding,Cookie,Authorization
7   last-modified: Sun, 05 Apr 2020 14:31:14 GMT
8   content-encoding: gzip
9   age: 10487
10  server-timing: cache;desc="hit-front"
11  strict-transport-security: max-age=106384710; includeSubDomains; preload
12  cache-control: private, s-maxage=0, max-age=0, must-revalidate
13  accept-ranges: bytes
14  content-length: 22883
15
16  <!DOCTYPE html>
17  <html class="client-nojs" lang="de" dir="ltr">
18  <head>
19  <meta charset="UTF-8"/>
20  <title>Weidengewächse - Wikipedia</title>
21  <link rel="stylesheet" href="/w/load.php?lang=de&amp;modules=ext.cite.styles%7Cext.flag
22  <script async="" src="/w/load.php?lang=de&amp;modules=startup&amp;only=scripts&amp;raw=
23  <meta name="ResourceLoaderDynamicStyles" content=""/>
24  <link rel="stylesheet" href="/w/load.php?lang=de&amp;modules=site.styles&amp;only=style
25  <meta name="generator" content="MediaWiki 1.35.0-wmf.28"/>
```

So-called HTTP response header

Oh! Look at that neat little HTML down here!

There's the text that shows up in the browser tab bar.

Example HTML document

```html
<!DOCTYPE html>
<html>
  <head>
    <title>My Little Pony</title>
    <meta name="description" content="Celebrate Rarity Month with My Little Pony! ..." />
    <link rel="stylesheet" href="/style.css" type="text/css" />
  </head>
  <body>
    <div class="main-content">
      <h1>Welcome to My Little Pony Online</h1>
      <p>Get to know the family of My Little Pony!</p>
      <hr />
      <h2>Meet the Rainbow Squads</h2>
      <p class="text-small">
        Whether an alicorn like Twilight Sparkle, a unicorn like Rarity, a pegasus like Ra
        an earth pony like Pinkie Pie and Applejack, get to know your favorite friends fro
        Girls!
      </p>
      <a href="/discover">Discover My Little Pony!</a>
    </div>
  </body>
</html>
```

… consists of nested elements
… elements are declared using tags

* **div:** diverse element
* **h1, h2, …:** headings
* **p:** paragraph
* **a:** anchor (aka a link)
* **head:** document meta data
* **body:** what is shown in the browser window

… elements may also have attributes (e.g. name, rel, class, …)


Oh my gosh....
I'm so FREAKING excited!!!

Step back!
So what do we actually want to do?

1. Request a web page from a web server
2. Receive the HTML response
3. Store the HTML response somewhere
4. Extract information from the HTML
5. Store extracted information in a useful format, e.g. Excel or CSV
6. Automate the process for a scheduled routine
7. Use extracted information to do something
   e.g. analyze it, trigger actions from it, …

Step back!
So what do we actually want to do?

1. Request a web page from a web server
2. Receive the HTML response
3. Store the HTML response somewhere
4. Extract information from the HTML
5. Store extracted information in a useful format, e.g. Excel or CSV
6. Automate the process for a scheduled routine
7. Use extracted information to do something
   e.g. analyze it, trigger actions from it, …

D. Kriesel:
„Rohdaten sind geil!“

Appendix A1

OSI Model

	Layer	Function	Example Protocols
7	Application Layer	network process to application	HTTP, SFTP, SSH
6	Presentation Layer	data representation & encryption	XML, JSON
5	Session Layer	interhost communication	Mostly theoretical
4	Transport Layer	end-to-end connections & reliability	TCP, UDP
3	Network Layer	path determination & logical addressing	IP Addresses
2	Data Link Layer	physical addressing	MAC Addresses
1	Physical Layer	physical media transmission	Ethernet, Bluetooth, Wireless

Source: https://www.wordfence.com/learn/understanding-the-osi-model-video/