Fundamentals of Web Development

Third Edition by Randy Connolly and Ricardo Hoar

Chapter 2

How the Web Works

In this chapter you will learn . . .

- **The fundamental protocols that make the web possible**
- **How the domain name system works**
- **Why HTTP is more than just a four-letter abbreviation**
- **How browsers and servers work to exchange and interpret HTML**

Internet protocols

- A **protocol** is a set of rules that partners use when they communicate.
- TCP/IP, from Chapter 1, is an essential internet protocol!
- These protocols have been implemented in every operating system and make fast web development possible. If web developers had to keep track of packet routing, transmission details, domain resolution, checksums, and more, it would be hard to get around to the matter of actually building websites.

Starting Note

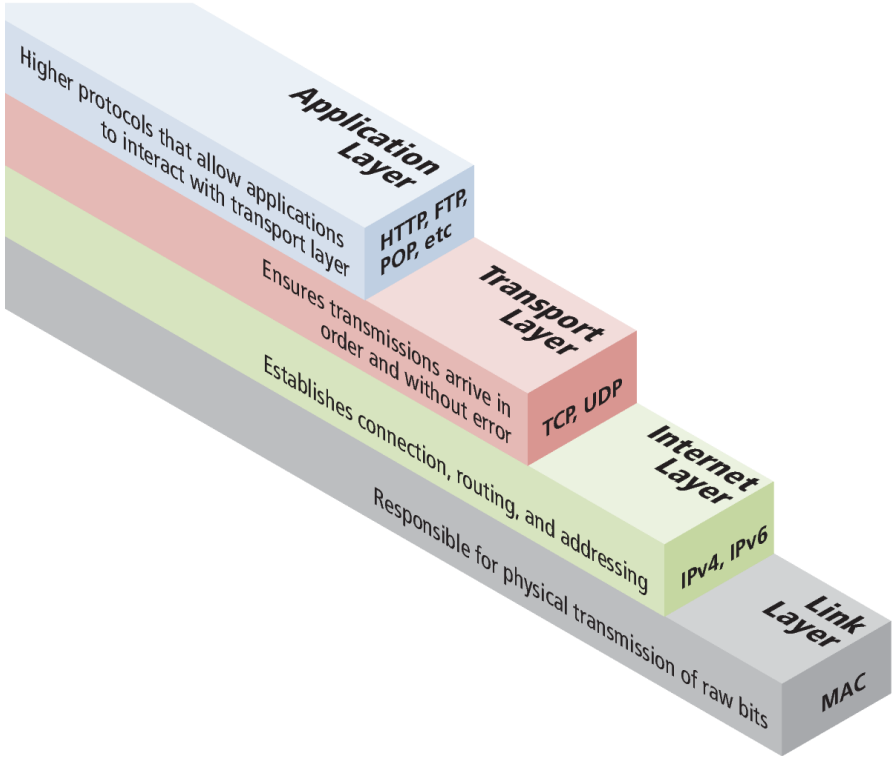
NOTE

Knowledge of how the web works, from low-level protocol to high-level JavaScript library, creates better web developers, which is why we start with some fundamental concepts in these early chapters.

There is a trend in web development to encourage web developers and designers to embrace this blending of roles as part of a holistic *DevOps* approach, which we describe in Chapter 17.

This means even if you're hired primarily to style CSS, you may need to know about HTML, IP addresses, domain names, web servers, browsers and more. Thankfully, you can always come back and revisit this material later when it's referenced again.

A Layered Architecture



- The TCP/IP Internet protocols were originally abstracted as a four-layer stack
- Later abstractions subdivide it further into five or seven layers
- Since we focus on the top layer, we will use the earliest and simplest **four-layer network model**.

Link Layer

- The **link layer** is the lowest layer, responsible for both the physical transmission of data across media and establishing logical links.
- It handles issues like packet creation, transmission, reception, error detection, collisions, line sharing, and more.
- One term that is sometimes used in the Internet context is that of **MAC** (media access control) **addresses**.

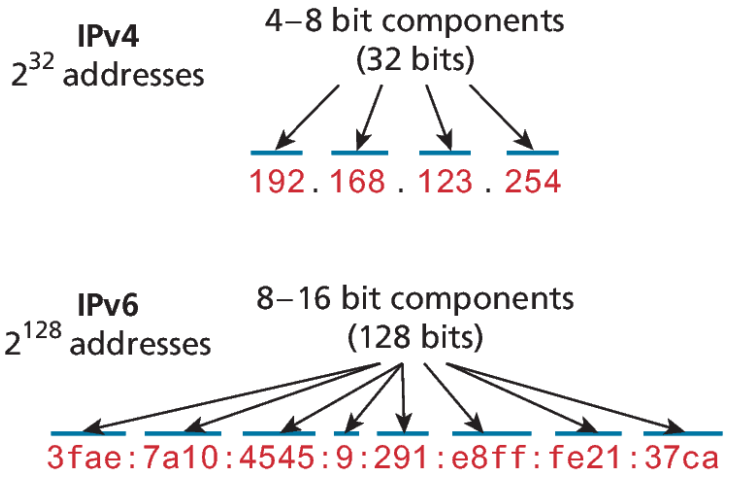
Internet Layer

- The **Internet layer** (sometimes also called the IP Layer) routes packets between communication partners across networks.
- It provides “best effort” communication. It sends out a message to its destination but expects no reply and provides no guarantee the message will arrive intact, or at all.
- The Internet uses the **Internet Protocol (IP) addresses**, which are numeric codes that uniquely identify destinations on the Internet.
- Every device connected to the Internet has such an **IP address**.

IP Addresses (cont)

There are two types of IP addresses: IPv4 and IPv6.

- In **IPv4**, four 8-bit integers separated by . encode the address.
- **IPv6** uses eight 16-bit integers and has over a billion billion times the number in IPv4



Port Address Translation (PAT)

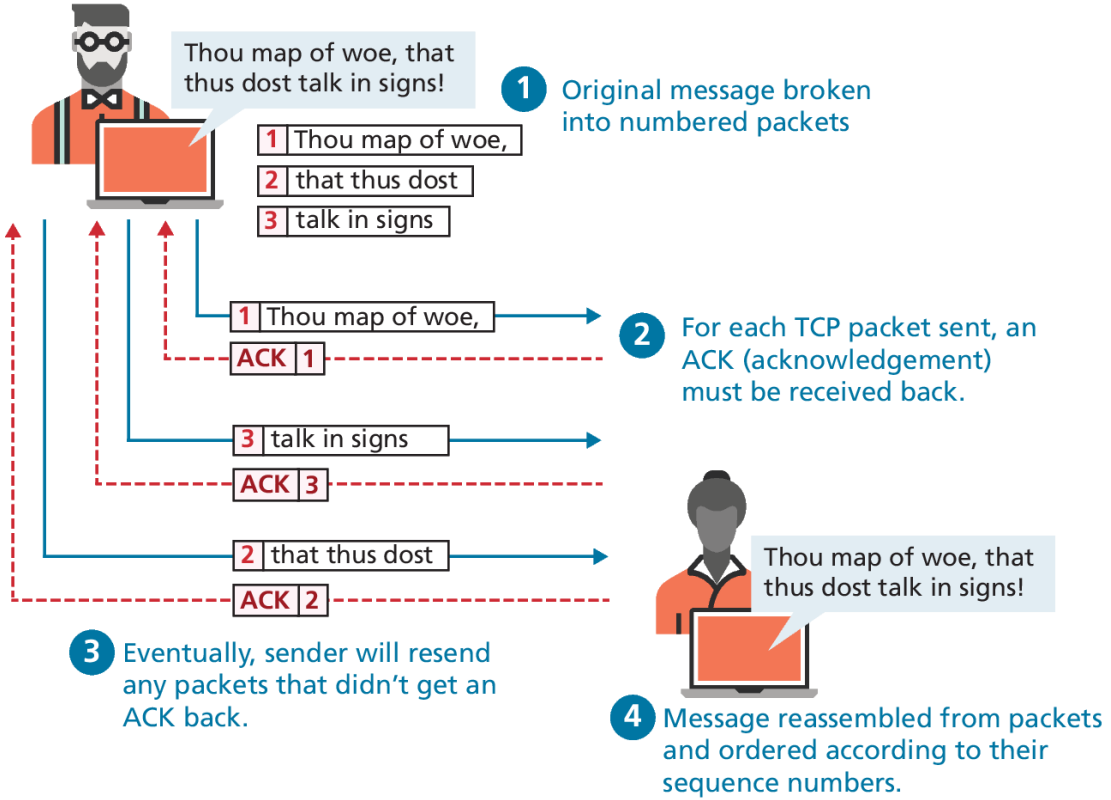
The IPv4 address space was depleted in 2011, but the number of computers connected to the Internet continued to grow.

- **Port Address Translation (PAT)**, allows multiple, unrelated networks to make use of the same IP address
- When you join a wireless network in a coffee shop, home, office or university, it is quite likely you are making use of PAT.
- For future growth, IPv6 will be necessary.

Transport Layer

- The **transport layer** ensures transmissions arrive in order and without error.
- First, the data is broken into **packets** formatted according to the **Transmission Control Protocol (TCP)**.
  - Each data packet has a header that includes a sequence number, so the receiver can put the original message back in order
  - Each packet acknowledges its successful arrival back to the sender (ACK).
  - In the event of a lost packet (since no ACK arrived for that packetthe packet will be retransmitted.
- This means you have **a *guarantee*** that messages sent will arrive and will be in order.

Transport Layer (example)



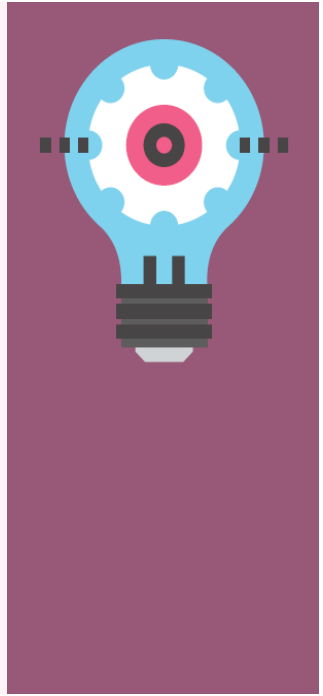
User Datagram Protocol (UDP)

PROTIP

Sometimes we do not want guaranteed transmission of packets.

Consider a live multicast of a soccer game, for example. Millions of subscribers may be streaming the game, and the broadcaster can't afford to track and retransmit every lost packet. A small loss of data in the feed is acceptable, and the customers will still see the game.

An Internet protocol called **User Datagram Protocol (UDP)** is used in these scenarios in lieu of TCP. Other examples of UDP services include Voice Over IP (VoIP), many online games, and Domain Name System (DNS).



Application Layer

- The **application layer** is the level of protocols familiar to most web developers.
- Application layer protocols implement process-to-process communication.
- There are many application layer protocols. A few that are useful to web developers include:
  - **HTTP.** The Hypertext Transfer Protocol is used for web communication.
  - **SSH**. The Secure Shell Protocol allows remote command-line connections to servers.
  - **FTP**. The File Transfer Protocol is used for transferring files between computers.
  - **POP/IMAP/SMTP.** Email-related protocols for transferring and storing email.
  - **DNS.** The Domain Name System protocol used for resolving domain names to IP addresses.

Domain Name System

- As elegant as IP addresses may be, human beings do not enjoy having to recall long strings of numbers.
- Even as far back as the days of ARPANET, researchers assigned **domain names** to IP addresses
- In those early days, the number of Internet hosts was small, so a list of a domains and associated IP addresses could be downloaded as needed as a **hosts** file (see Pro Tip p51).
- As the number of computers on the Internet grew, this **hosts** file had to be replaced with a better, more scalable, and distributed system. This system is called the **Domain Name System (DNS)**

DNS Overview

- The DNS system maps resolves domain names to IP addresses.
- By separating the domain name of a server from its IP address, a site can move to a different host without changing its name.
- Since the entire request-response cycle can take less than a second, it is easy to forget that DNS requests are happening in all.
- The actual process is more complex (more on that later)

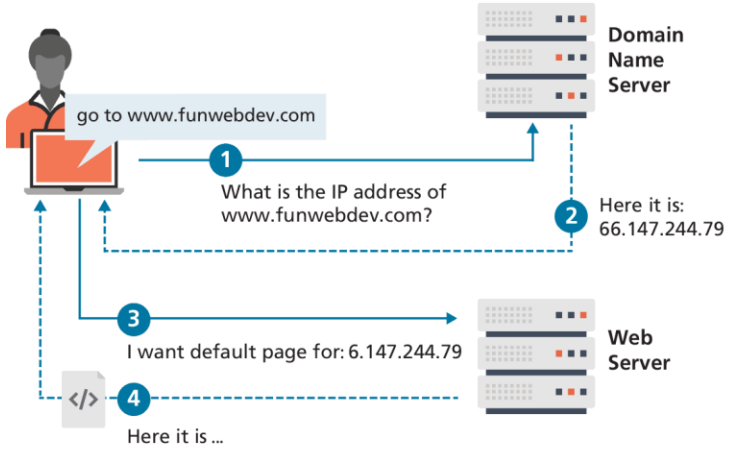
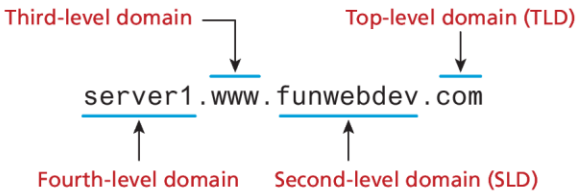
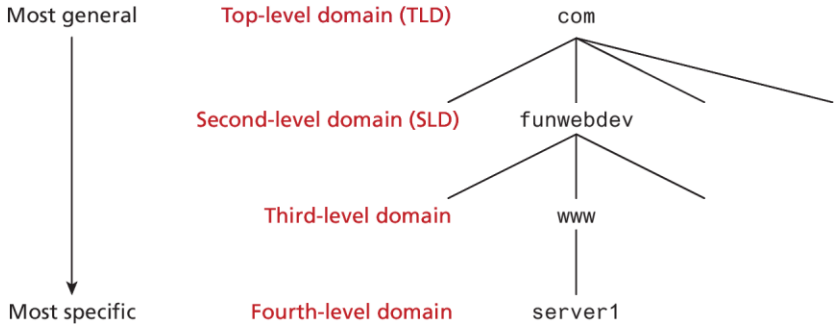


FIGURE 2.6 DNS overview (simplified)

Name Levels

- A domain name can be broken down into several parts, which describe a hierarchy.
- All domain names have at least a **top-level domain (TLD)** name and a **second-level domain (SLD)** name





Name Levels (Top Level)

The rightmost portion of the domain name (to the right of the rightmost period) is called **the top-level domain**. For the top level of a domain, we are limited to two broad categories, plus a third reserved for other use.

1. **Generic top-level domain (gTLD)**
2. **Country code top-level domain (ccTLD)**
3. **.arpa** (used for reverse DNS lookups)

Generic top-level domain (gTLD)

Generic top-level domains (gTLD) include the famous .com and ,org. There are 3 subtypes of gTLD.

- **Unrestricted.** TLDs include **.com**, **.net**, **.org**, and **.info**.
- **Sponsored.** TLDs including **.gov**, **.mil**, **.edu**, and others
- **New.** Starting in June 2012, ICANN invited companies to launch new TLDs in order to provide more choice. Since then over 1000 new TLD have been created including .art, .cash, .cool, .jobs, .tax and so on

Country code top-level domain

Country code top-level domain (ccTLD) are under the control of the countries which they represent, which is why each is administered differently.

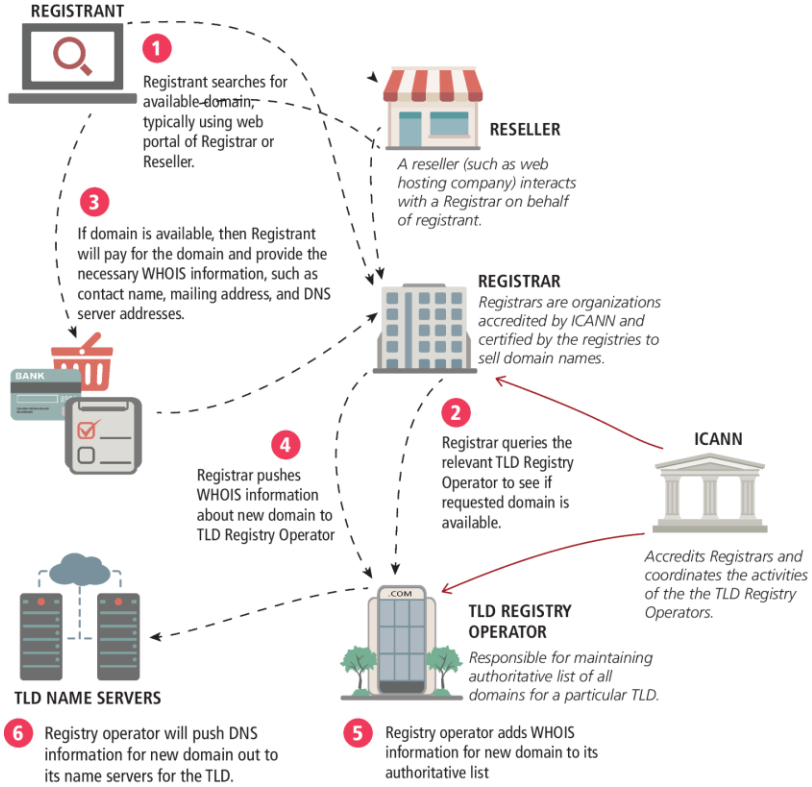
- In the United Kingdom, for example, businesses must register subdomains to **co.uk** rather than second-level domains directly whereas in Canada, **.ca** domains can be obtained by any person, company, or organization living or doing business in Canada.
- Other countries have peculiar extensions with commercial viability (such as **.tv** for Tuvalu) and have begun allowing unrestricted use to generate revenue.
- **Internationalized top-level domain name (IDN)** allows domains to use non-ascii characters and has been deployed since 2009. There are over 9 million IDN domains

Name Registration

- Q: How then are domain names assigned?
- A: Special organizations or companies called **domain name registrars** manage the registration of domain names. These domain name registrars are given permission to do so by the appropriate generic top-level domain (gTLD) registry and/or a country code top-level domain (ccTLD) registry.
- The nonprofit **Internet Corporation for Assigned Names and Numbers (ICANN)**—oversees the management of toplevel domains, accredits registrars, and coordinates other aspects of DNS.

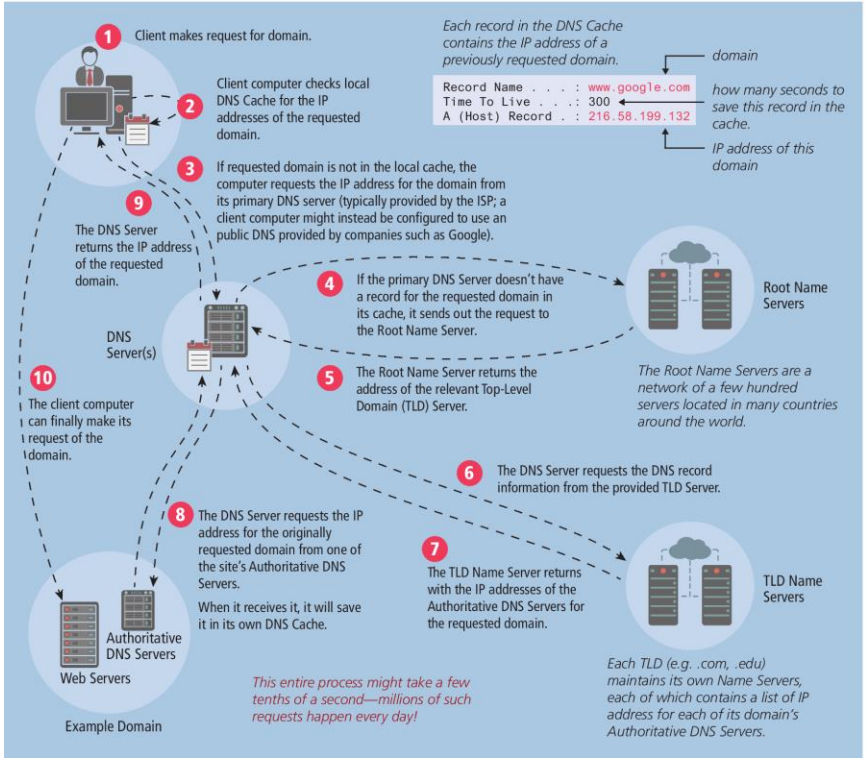
Domain name registration process

1. Registrant searches for domain, typically using web portal of Registrar or Reseller.
2. Registrar queries the relevant TLD Registry Operator to see if requested domain is available.
3. If domain is available, then Registrant will pay for the domain and provide the necessary WHOIS information
4. Registrar pushes WHOIS information about new domain to TLD Registry Operator
5. Registry operator adds WHOIS information for new domain to its authoritative list
6. Registry operator will push DNS information for new domain out to its name servers for the TLD.



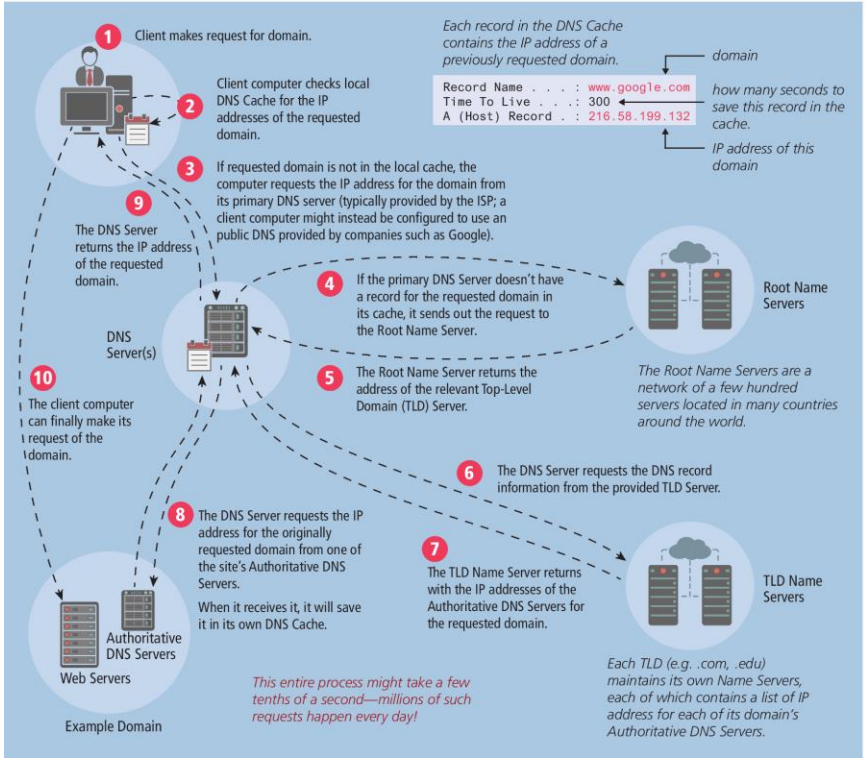
Address Resolution

1. Client makes request for domain
2. Client computer checks local DNS Cache
3. If requested domain is not in the local cache, the computer requests the IP address for the domain from its primary DNS server
4. If the primary DNS Server doesn't have a record for the requested domain in its cache, it sends out the request to the Root Name Server
5. The Root Name Server returns the address of the relevant Top-Level Domain (TLD) Server.



Address Resolution (cont)

6. The DNS Server requests the DNS record information from the provided TLD Server.
7. The TLD Name Server returns with the IP addresses of the Authoritative DNS Servers for the requested domain.
8. The DNS Server requests the IP address for the originally requested domain from one of the site's Authoritative DNS Servers.
9. The DNS Server returns the IP address of the requested domain.
10. The client computer can finally make its request of the domain.



Uniform Resource Locators (optional)

Optional components of the URL are:

- the **path** (which identifies a file or directory to access on that server),
- the **port** to connect to,
- a **query string**, and
- a **fragment** identifier

Port (URL)

- A **port** is a type of software connection point used by the underlying TCP/IP protocol and the connecting computer.
- Although the port attribute is not commonly used in production sites, it can be used to route requests to a test server, to perform a stress test, or even to circumvent Internet filters.
- If no port is specified, the protocol determines which port to use. For instance, port 80 is the default port for web-related HTTP requests.
- Syntax is to add a colon after the domain, then specify an integer port number. **http://funwebdev.com:8080/** would connect on port 8080

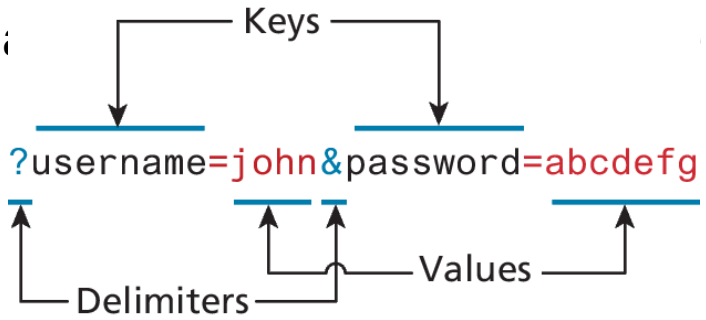
Path (URL)

- The **path** is an important concept to anyone who has ever used a computer file system.
- The root of a web server corresponds to a folder somewhere on that server.On many Linux servers that path is **/var/www/html/** or something similar (for Windows it is often **/inetpub/wwwroot/**).
- The path is optional. However, when requesting a folder or the top-level page of a domain, the web server will decide which file to send you.

Query String (URL)

Query strings will be covered in depth when we learn more about HTML forms and server-side programming. They are a critical way of passing information, such as user form input, from the client to the server.

- In URLs, they are encoded as key-value pairs delimited by & symbols and preceded by the ? Symbol
- An example query string for passing name a
  2.11



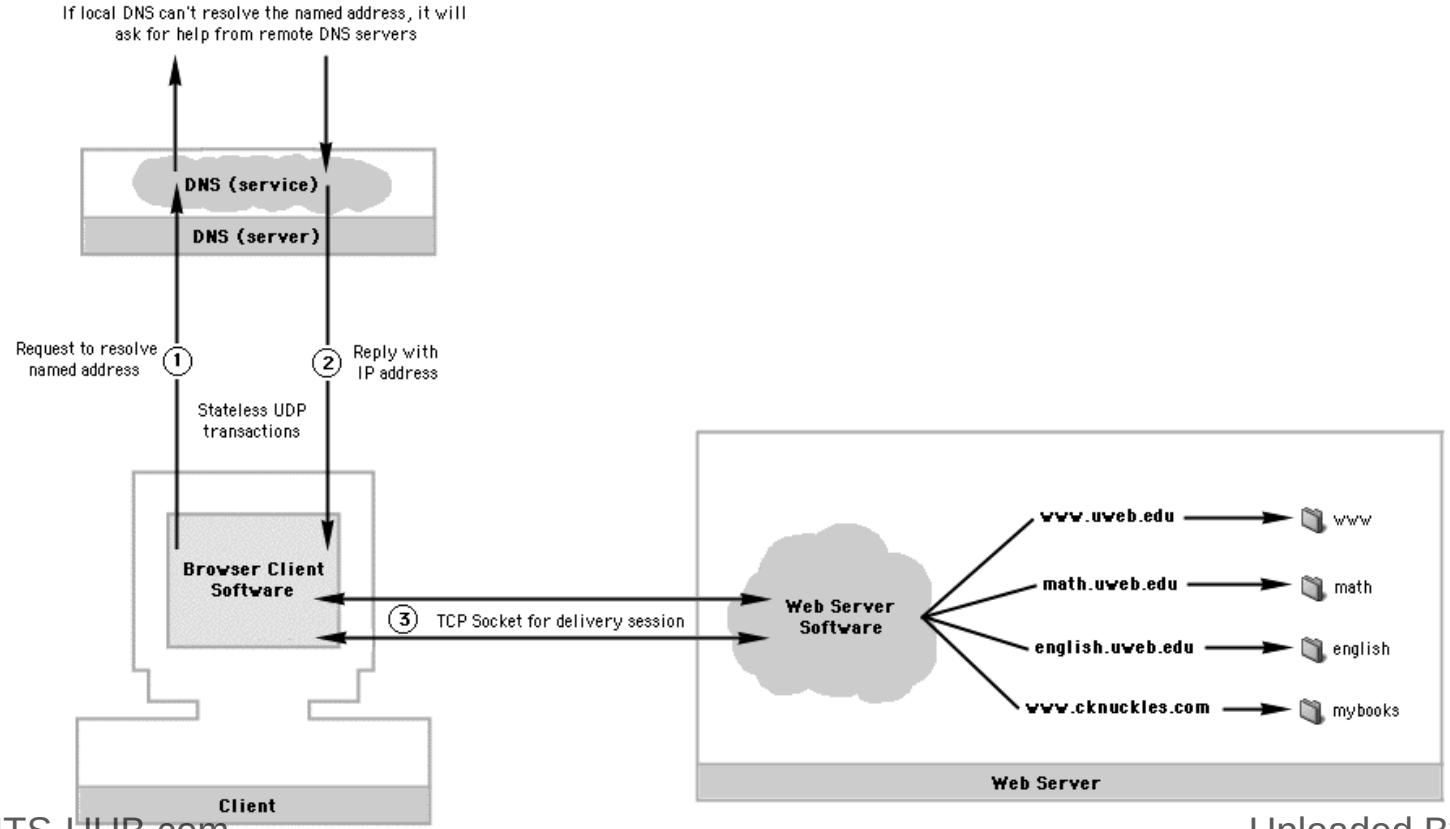
Fragment (URL)

- The last part of a URL is the optional fragment.
- This is used as a way of requesting a portion of a page.
- Browsers will see the fragment in the URL (denoted by #), seek out the fragment tag anchor in the HTML, and scroll the website down to it.
- “back to top” links are a common use of fragments.

Two useful application layer services on the Web.

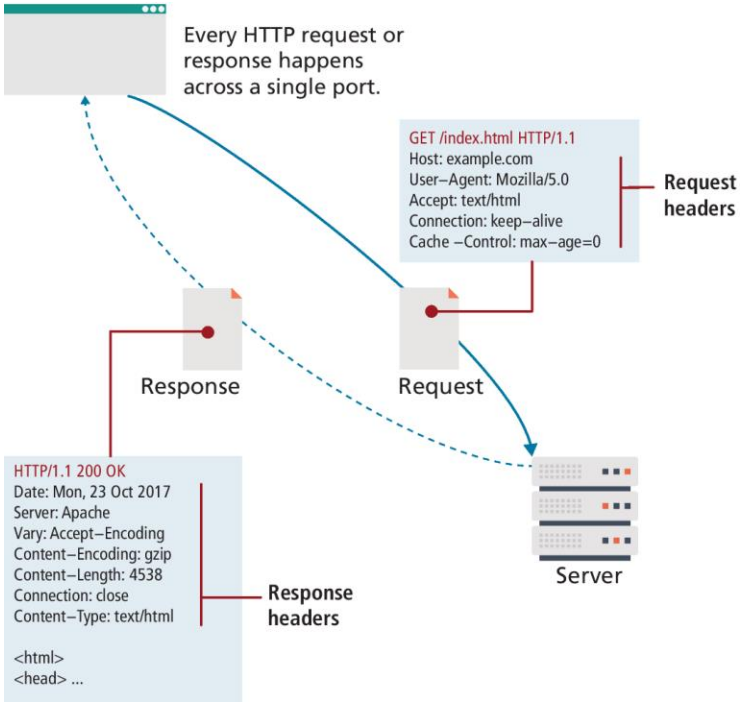
Domain Name Service (DNS) -- Uses UDP for transport.

Virtual Hosting -- Maps Domain Names onto folders on the Web server.



Hypertext Transfer Protocol

- **HTTP** is an essential part of the web.
- HTTP establishes a TCP connection on port 80 (by default). The server waits for the request, and then responds with a
  - Headers,
  - Response code,
  - an optional message (which can include files)



HTTP Headers

Headers are sent in the request from the client and received in the response from the server. Headers are one of the most powerful aspects of HTTP and unfortunately, few developers spend any time learning about them.

- **Request headers** include data about the client machine
  - Host, User-Agent, Cache settings and more
- **Response headers** have information about the server answering the request and the data being sent
  - Server, Last Modified, Content Type, Encoding,

HTTP Request Methods

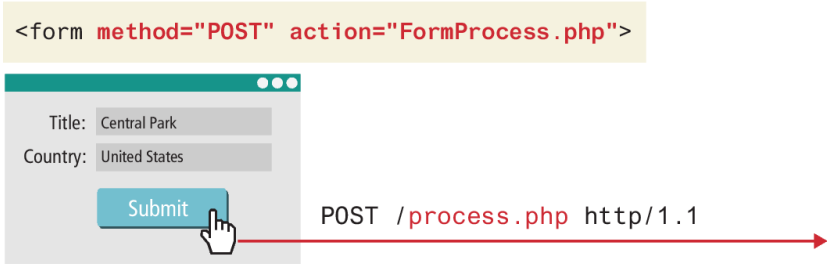
- The most common requests are the GET and POST request, along with the HEAD request
- In Chapter 13 you will make use of the PUT and DELETE requests when creating an API in Node.
- Other HTTP verbs such as CONNECT, TRACE, and OPTIONS are less commonly used and are not covered in the book.

GET Request

- The most common type of HTTP request is the **GET request**.
- One is asking for a resource located at a specified URL to be retrieved.
- Whenever you click on a link, type in a URL in your browser, or click on a bookmark, you are *usually* making a GET request.
- Data can be transmitted through a GET request, with a query string

POST Request

- The other common request method is the **POST request**.
- This method is normally used to transmit data to the server using an HTML form



Response Codes

- **Response codes** are integer values returned by the server as part of the response header.
- These codes describe the state of the request, including whether it was successful, had errors, requires permission, and more.
- The codes use the first digit to indicate the category of response.
  - 2## codes are for successful responses,
  - 3## are for redirection-related responses,
  - 4## codes are client errors, while
  - 5## codes are server errors.

HTTP Response Codes (Table 2.1 edited)

Code	Description
200: OK	The request was successful.
301: Moved Permanently	Tells the client that the requested resource has permanently moved.
304: Not Modified	If the client requested a resource with appropriate Cache-Control headers, the response might say that the resource on the server is no newer than the one in the client cache.
401: Unauthorized	Some web resources are protected and require the user to provide credentials to access the resource.
404: Not found	404 codes are one of the only ones known to web users. Many browsers will display an HTML page with the 404 code to them when the requested resource was not found.
414: Request URI too long	A 414 response code likely means too much data is likely trying to be submitted via the URL.
500: Internal server error	This error provides almost no information to the client except to say the server has encountered an error.

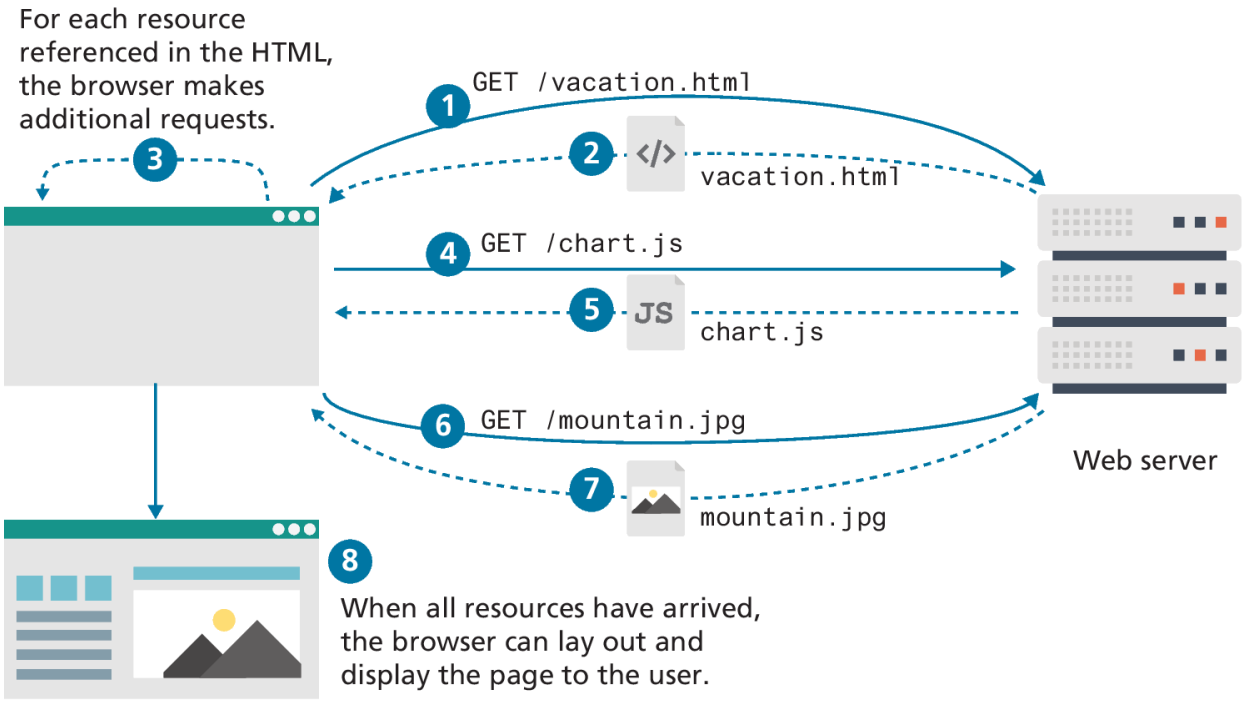
Web Browsers

- The user experience for a website is unlike the user experience for traditional desktop software.
- Users do not download software; they visit a URL, which results in a web page being displayed.
- Although a typical web developer might not build a browser, or develop a plugin, they must understand the browser's crucial role in web development.

Web Browsers (Fetching a web page)

- Seeing a single web page is facilitated by the browser, which
  - requests the initial HTML page, then
  - parses the returned HTML to find all the resources referenced from within it (like images, style sheets, and scripts).
- Only when all the files have been retrieved is the page fully loaded for the user
- A single web page can reference dozens of files and requires many HTTP requests and responses.

Fetching a web page diagram

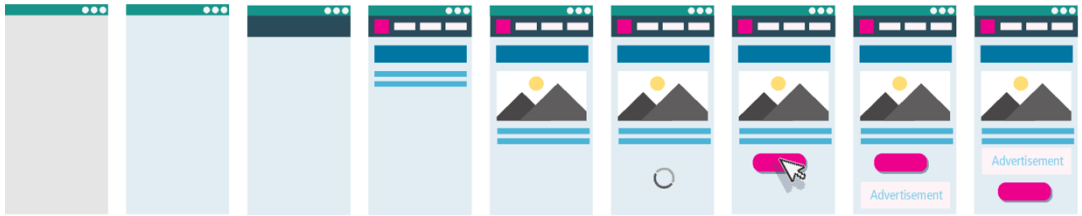


Browser Rendering

- The algorithms within browsers to download, parse, layout, fetch assets, and create the final interactive page for the user are commonly referred to collectively as the *rendering* of the page
- We will focus on the browser-rendering process through a user-centric lens where measures are categorized around perceived loading performance, interactivity, and visual stability

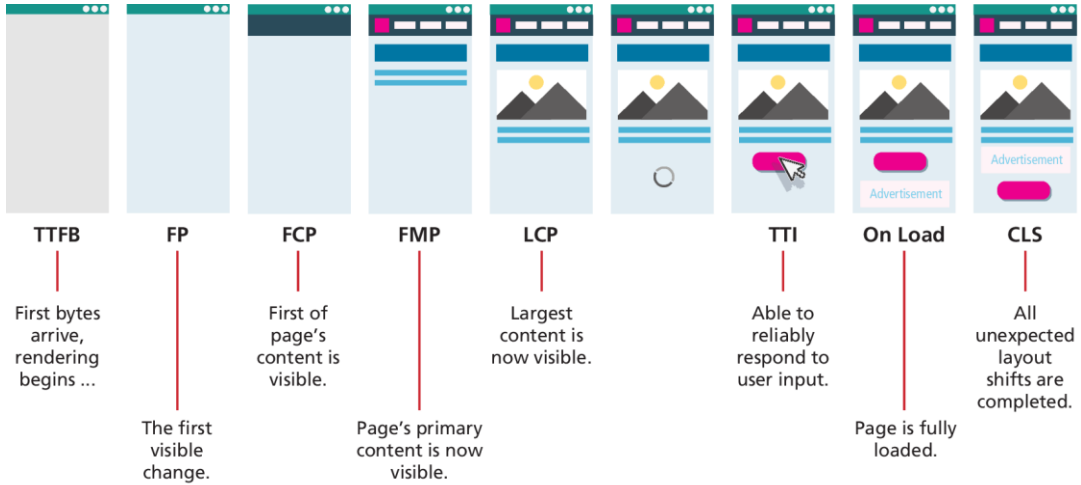
Browser Rendering (ii)

- The algorithms within browsers to download, parse, layout, fetch assets, and create the final interactive page for the user are commonly referred to collectively as the *rendering* of the page
- We will focus on the browser-rendering process through a user-centric lens where measures are categorized around perceived loading performance, interactivity, and visual stability



Browser Rendering Performance

- Time to First Byte (TTFB)
- First Paint (FP)
- First Contentful Paint (FCP)
- First Meaningful Paint (FMP)
- Largest Contentful Paint (LCP)
- Time to Interactive (TTI)
- On Load
- Cumulative Layout Shift (CLS)



Browser Caching

- Once a webpage has been downloaded from the server, it's possible that the user, a short time later, wants to see the same web page and refreshes the browser or rerequests the URL.
- Although some content might have changed, the majority of the referenced files are likely to be unchanged, so they needn't be redownloaded.
- Browser caching has a significant impact in reducing network traffic and will be come up gain in greater detail throughout this book.

Web Servers

- A **web server** is, at a fundamental level, nothing more than a computer that responds to HTTP requests.
- Real-world websites typically have many web servers configured together in web farms.
- Regardless of the physical characteristics of the server, one must choose an application stack to run a website. This **application stack** will include
  - an operating system,
  - web server software,
  - a database, and
  - a scripting language to process dynamic requests.

LAMP

- We will be using the **LAMP software stack**, which refers to the
  - **L**inux operating system,
  - **A**pache web server,
  - **M**ySQL database, and
  - **P**HP scripting language
- The Apple OSX MAMP software stack is nearly identical to LAMP, since OSX is a Unix implementation, and includes all the tools available in Linux.
- The WAMP software stack is another popular variation where Windows operating system is used.

Alternative Stacks

- Besides the LAMP stack, you will be using the **MERN stack** in the book, which refers to **M**ongoDB database, **E**xpress application framework, the JavaScript **R**eact framework, and **N**ode.js as the web server and execution environment.
- Many corporate intranets instead make use of the Microsoft **WISA software stack**, which refers to **W**indows operating system, **I**IS web server, **S**QL Server database, and the **A**SP.NET server-side development technologies.
- Another web development stack that is growing in popularity is the so-called **JAM stack**, which refers to **J**avaScript, **A**PIs, and **m**arkup.

Key Terms

address resolution

Apache

Application stack

application layer

country code top-level domain (ccTLD)

Cumulative Layout Shift (CLS)

DNS resolver

DNS server

domain names

domain name registrars

Domain Name System (DNS)

First Contentful Paint (FCP)

First Meaningful Paint (FMP)

First Paint (FP)

four-layer network model

generic top-level domain (gTLD)

GET request

google.com

HEAD request

Hypertext Transfer Protocol (HTTP)

Internet Corporation for Assigned Names and Numbers (ICANN)

Internet Assigned Numbers Authority (IANA)

internationalized top-leve domain name (IDN)

Internet layer

Internet Protocol (IP) addresses

IP address

IPv4

IPv6

Key Terms (cont)

JAM stack
Largest Contentful Paint
LAMP software stack
link layer
MAC addresses
MEAN software stack
On Load
packet
protocol
punycode
port
Port Address Translation
(PAT)
POST request
protocol
request
request headers
response codes
response headers
reverse DNS lookups
root name server
second-level domain
subdomain
Time to First Byte (TTFB)
Time to Interactive (TTI)
transport layer
Transmission Control Protocol (TCP)
top-level domain (TLD)
TLD name server
User Datagram Protocol (UDP)
Uniform Resource Locator (URL)
web server
WISA software stack

Copyright