<WA1/>

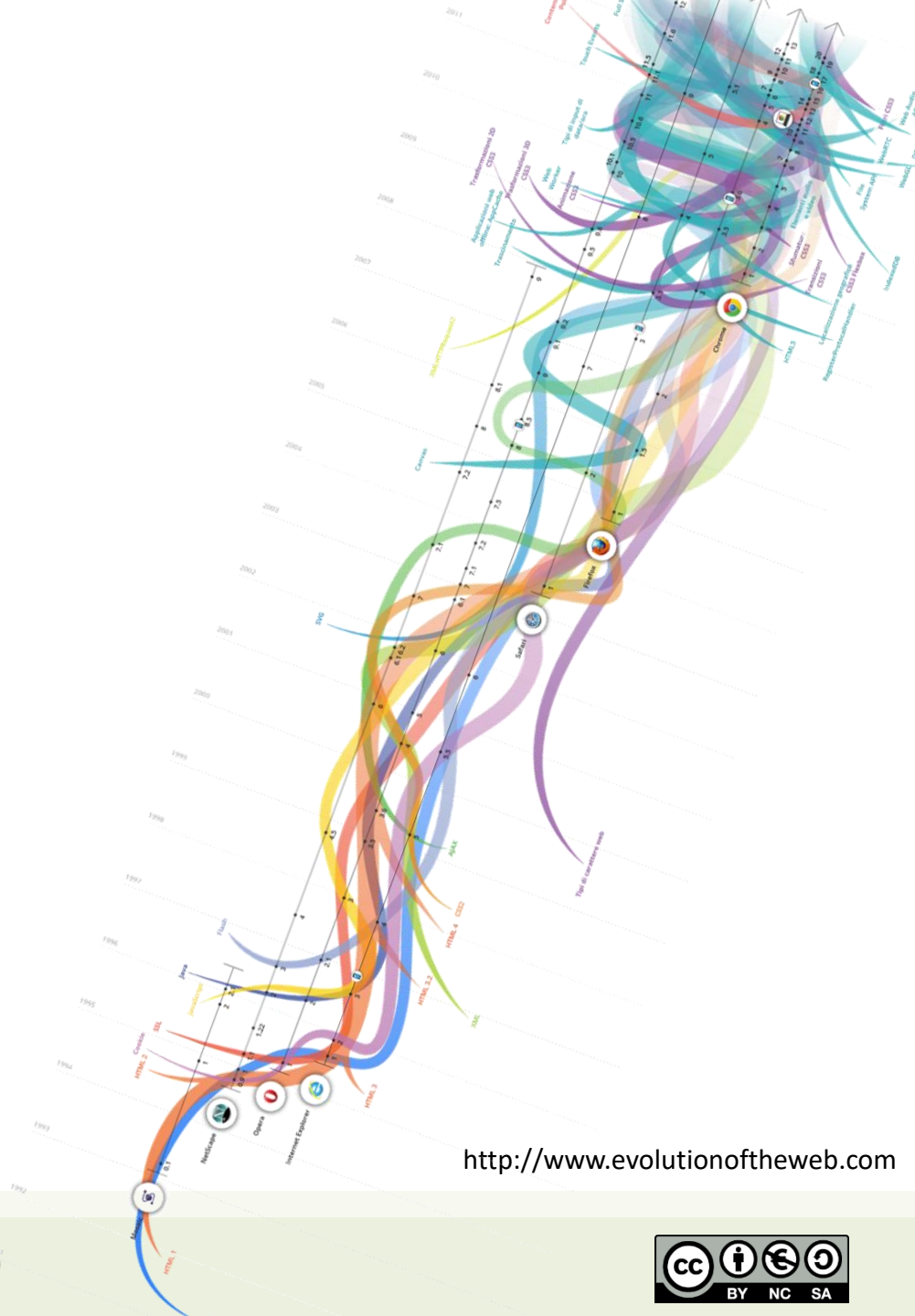
2020

Web Architecture

Layers, Languages, Protocols

Enrico Masala

Fulvio Corno

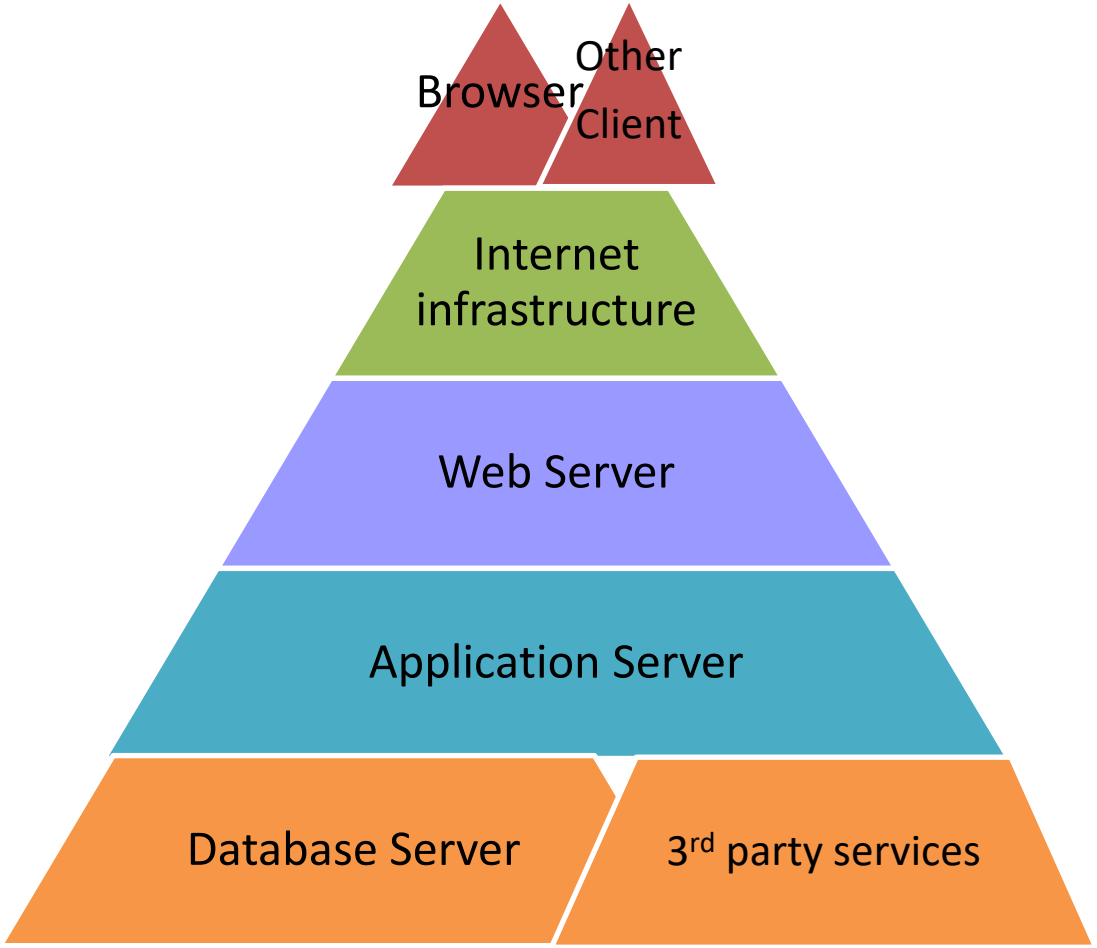


http://www.evolutionoftheweb.com

Goal

- Understand the architecture of the web
  - Main (software) components
  - Main network protocols
  - Main (programming|declarative) languages
- Standard vs. programmable components
- Interaction and communication across components

N-tier (N-level) architecture



- Each level/tier has a well defined role
- One or more servers implement each tier/layer
- More servers can share the same hardware or can run on dedicated devices
- Communication between tiers/levels is achieved through the network

General Architecture

- Historically, a web browser
- Also:
  - Mobile app
  - Desktop app
  - Other server application

General Architecture

- Web Server(s)
- Application Server(s)
- Database Server(s)

- Authentication, encryption, streaming, … server(s)

General Architecture

- Web Server(s)
- Application Server(s)
- Database Server(s)

- Authentication, encryption, streaming, … server(s)

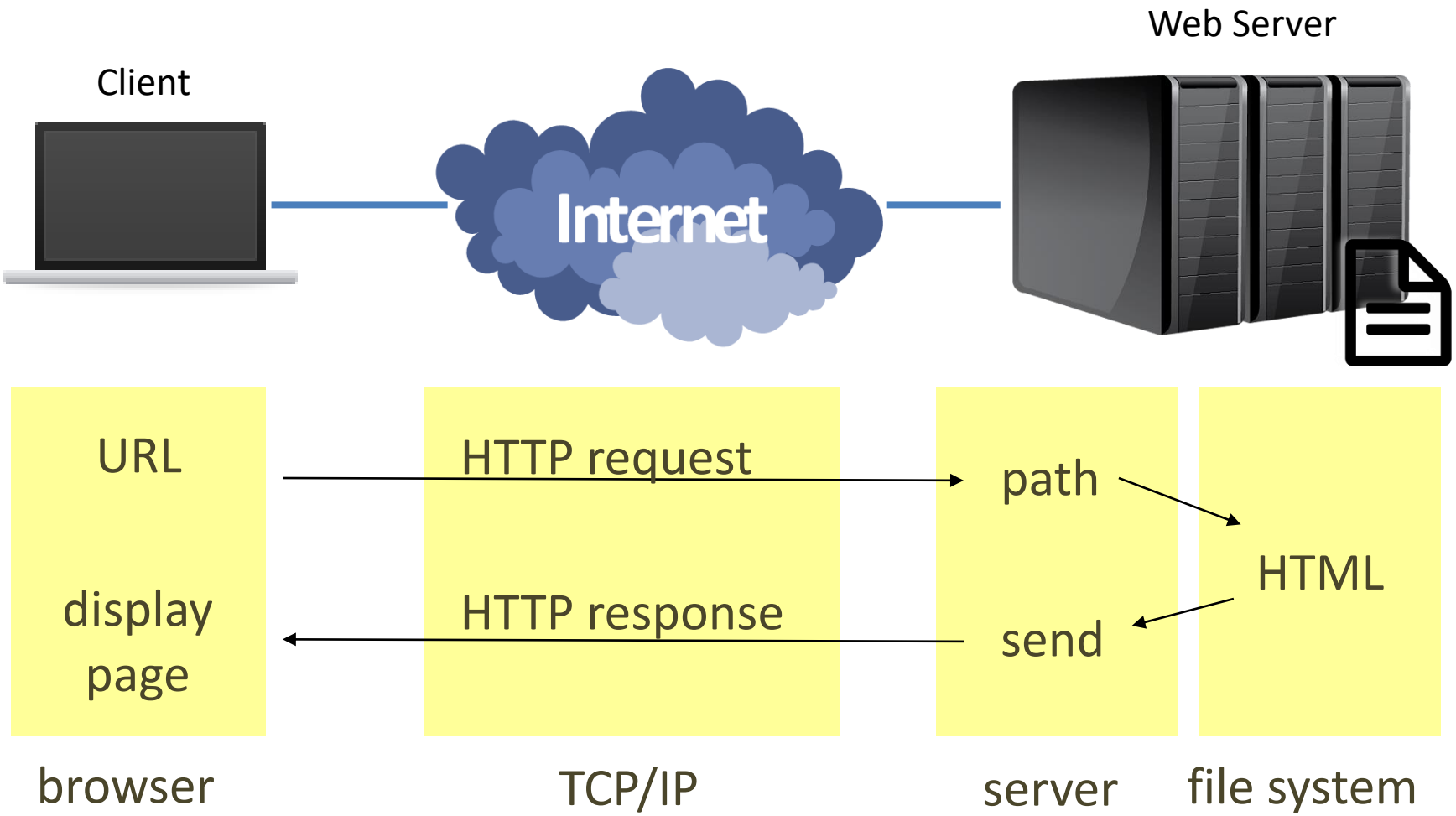
Definition

- “Server”may be defined as:
  - Logical definition:
    A process that runs on a host that relays information to a client upon the client sending it a request.
  - Physical definition:
    A host computer on a network that holds information (e.g., Web sites) and responds to requests for information

Web server

- Manages the HTTP protocol (handles requests and provides responses)
  - Receives client requests
  - Reads static pages from the filesystem
  - Activates the application server for dynamic pages (server-side)
  - Provides an HTML file back to the client
- One HTTP connection for each request
- Multi-process, Multi-threaded or Process pool

Web server

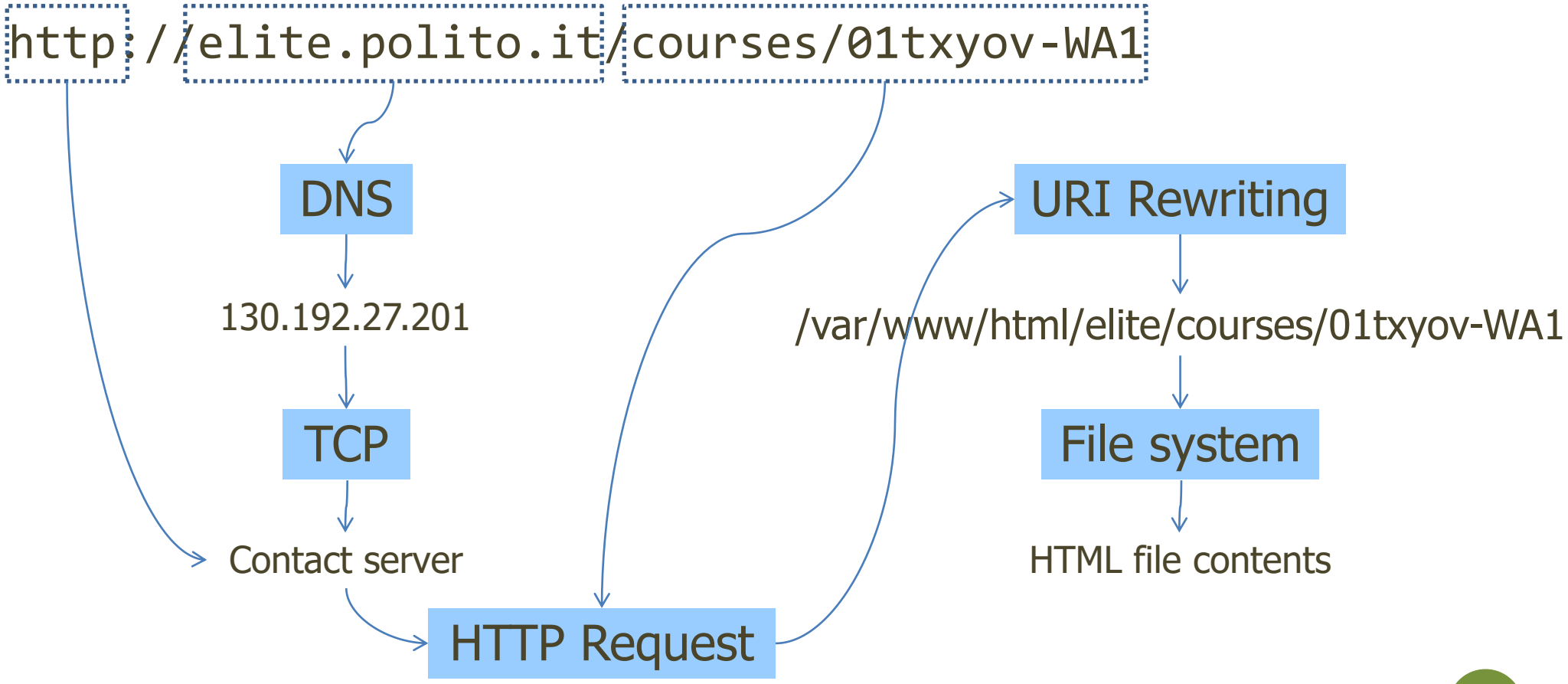

Web Server
Client
Internet
URL
HTTP request
path
HTML
display page
HTTP response
send
browser
TCP/IP
server
file system

Adopted standards

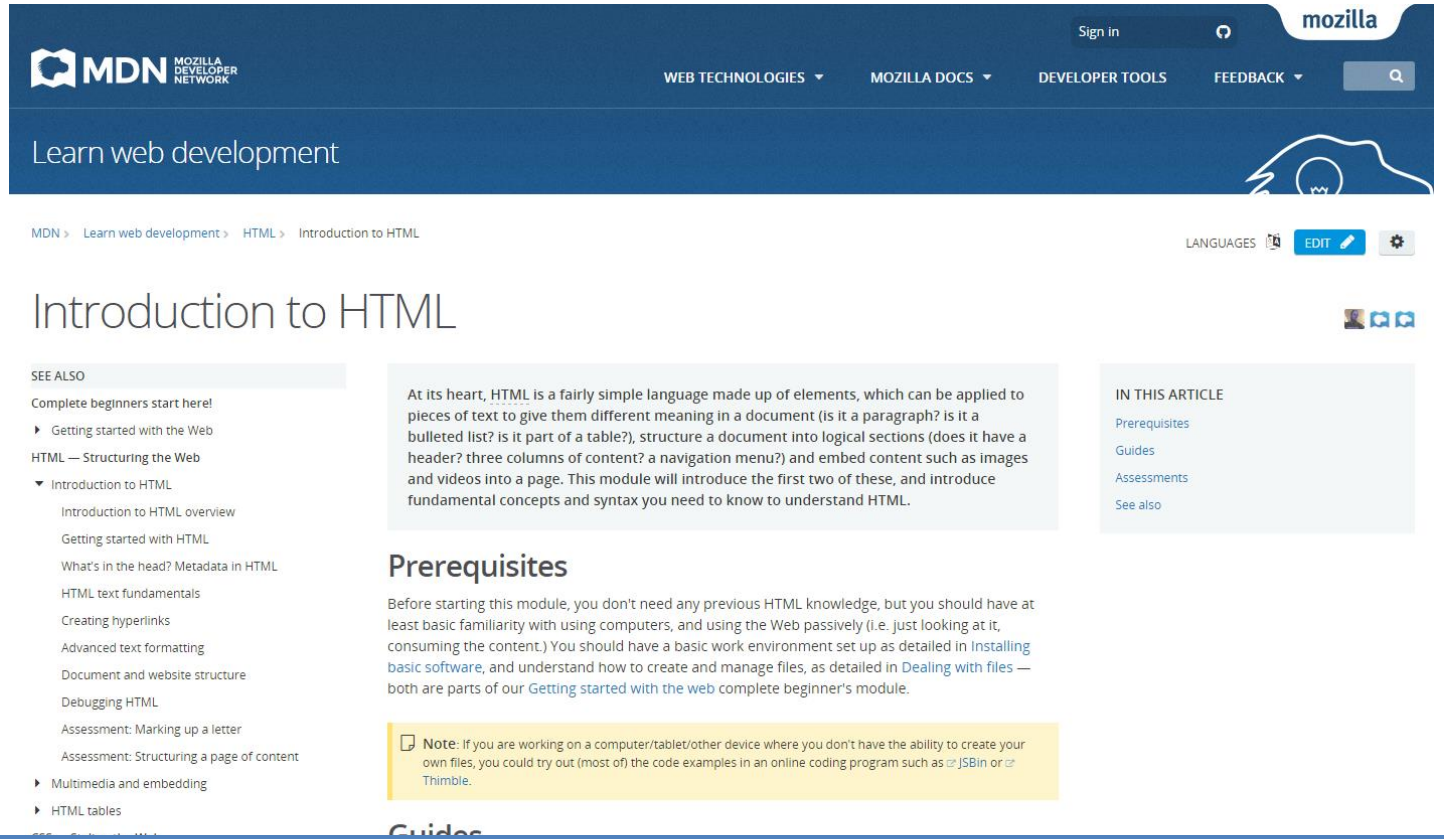
- URL (uniform resource locator) for finding web pages
- HTML (hyper text markup language) for writing web pages
- GIF (graphics interchange format) for images
- HTTP (hyper text transfer protocol) for client-server interaction
- TCP/IP (transmission control protocol over internet protocol) for data transfer

URL: Example

RFC 2396
http://www.w3.org/Addressing/

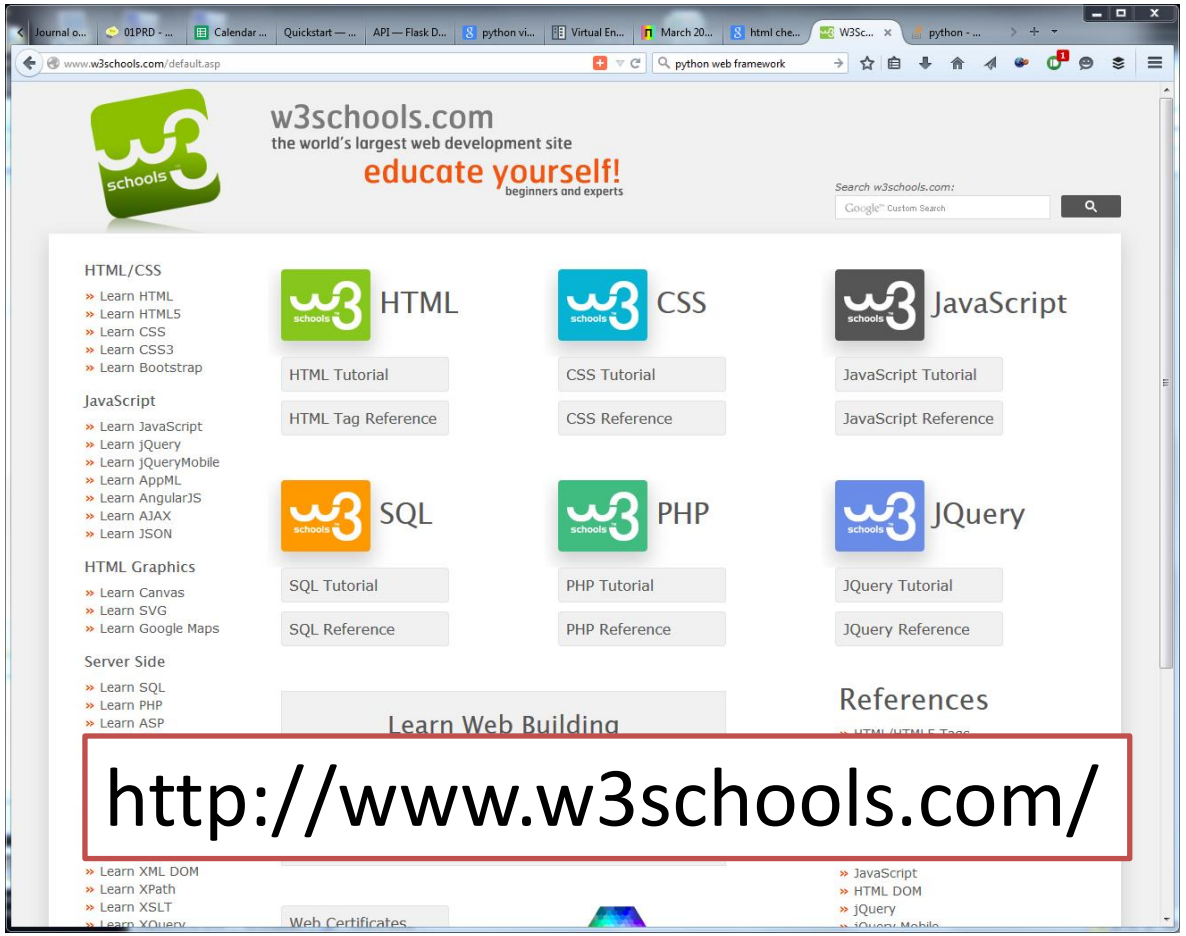


Getting started with HTML…



https://developer.mozilla.org/docs/Learn/HTML/Introduction_to_HTML

HTML in 5 minutes



URI Basics

- http://www.sadev.co.za/users/1/contact

- http://www.sadev.co.za?user=1&action=contact

- https://bbd.co.za/index.html#about

HTTP protocol

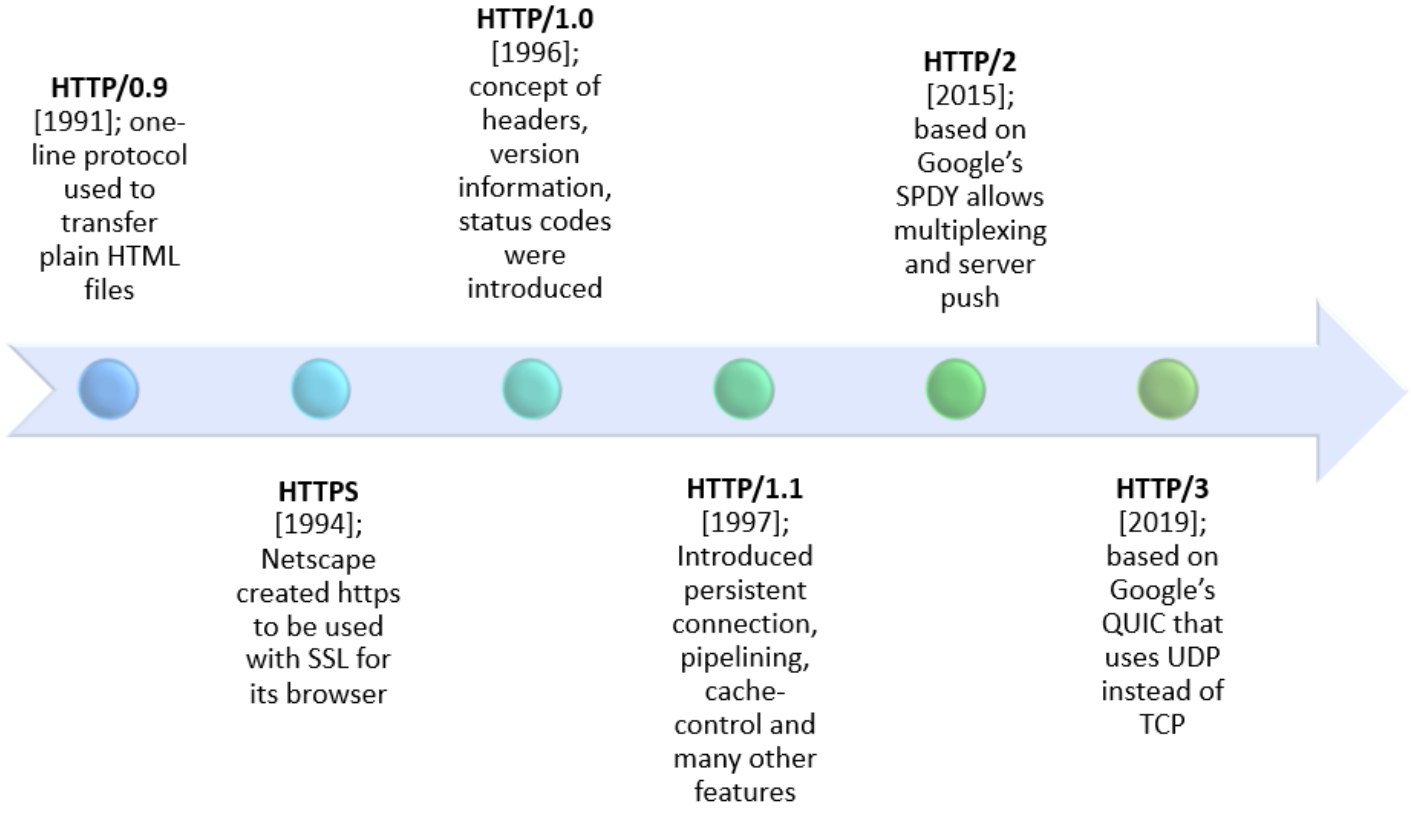
RFC 2616, RFC 2617
http://www.w3.org/Protocols

```
GET / HTTP/1.1
Host: elite.polito.it
User-Agent: Mozilla/5.0
Accept: text/html,appl
Accept-Language: it-I
Accept-Encoding: gzi
Cookie: __utma=1885
Connection: keep-a
```

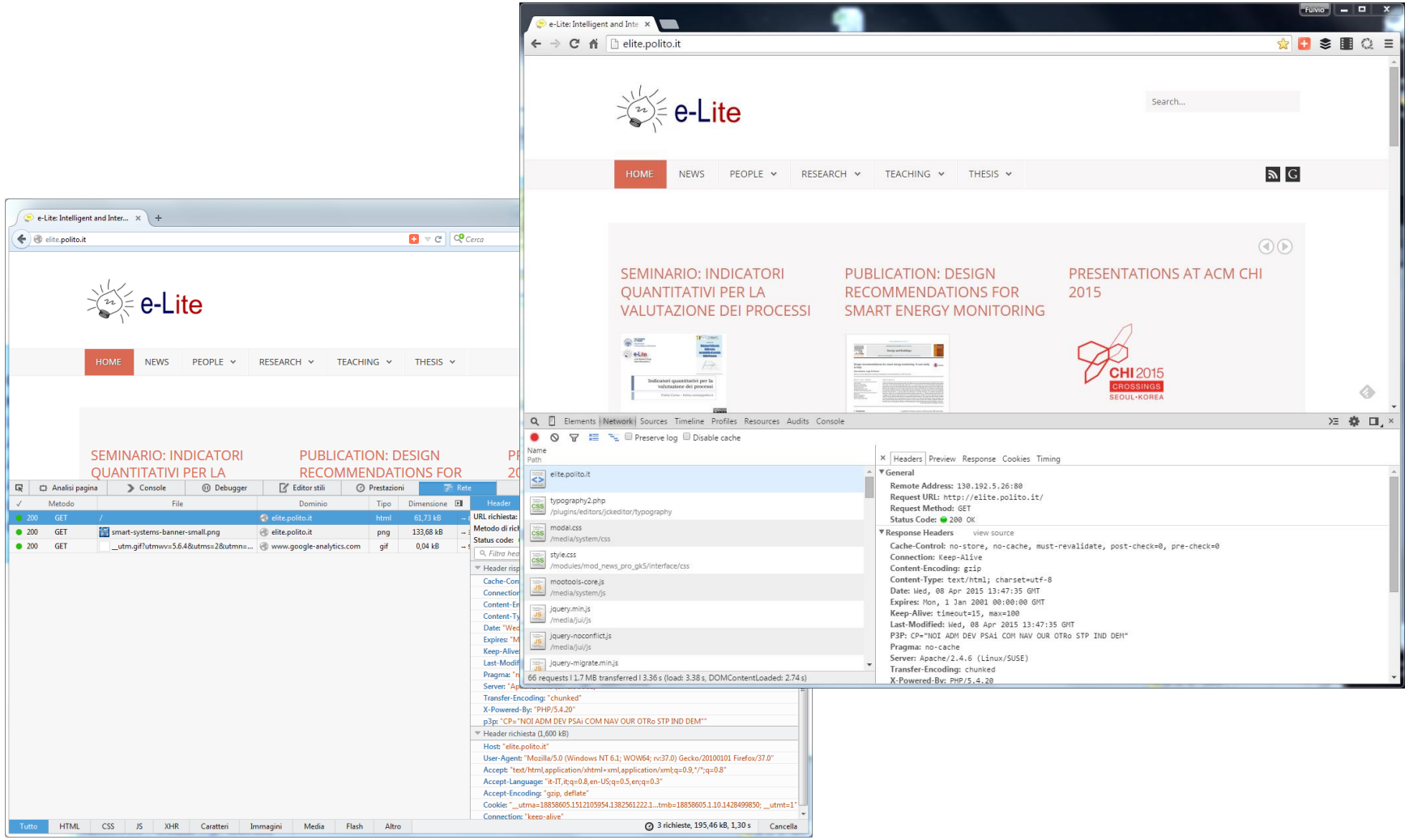
```
HTTP/1.0 200 OK
Cache-Control: no-store, no-cache, must-revalidate,
Connection: Keep-Alive
Content-Encoding: gzip
Content-Type: text/html; charset=utf-8
Date: Wed, 08 Apr 2016 13:36:24 GMT
Expires: Mon, 1 Jan 2020 00:00:00 GMT
Keep-Alive: timeout=15, max=100
Last-Modified: Wed, 08 Apr 2016 13:36:24 GMT
Pragma: no-cache
Server: Apache/2.4.6 (Linux/SUSE)
Transfer-Encoding: chunked
X-Powered-By: PHP/5.6.30
p3p: CP="NOI ADM DEV PSAi COM NAV OUR OTRo STP IND DEM«

<!DOCTYPE html>
<html>
<head>
. . . . . .
```

HTTP evolution



Browser developer tools

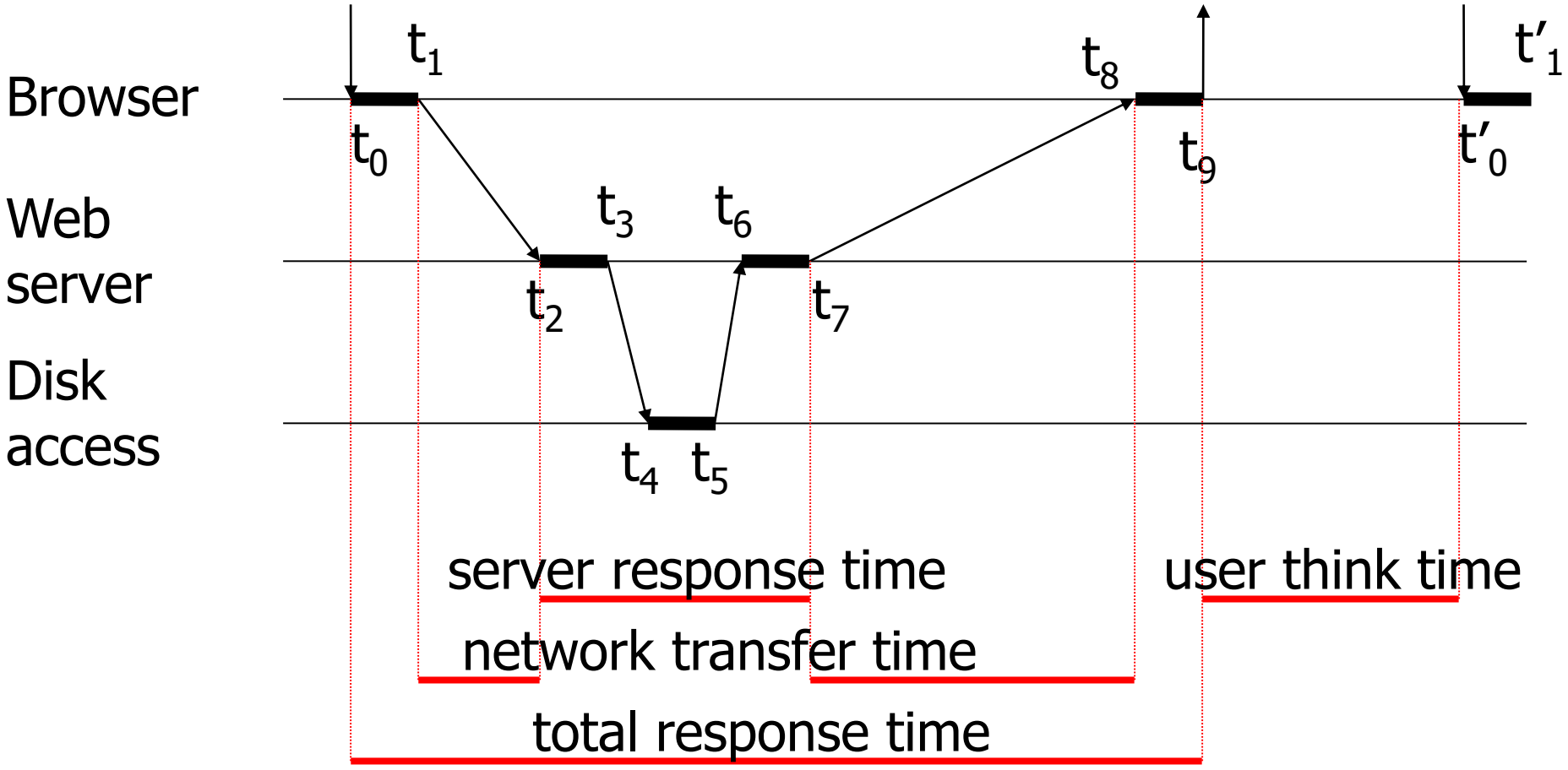


Performance measures

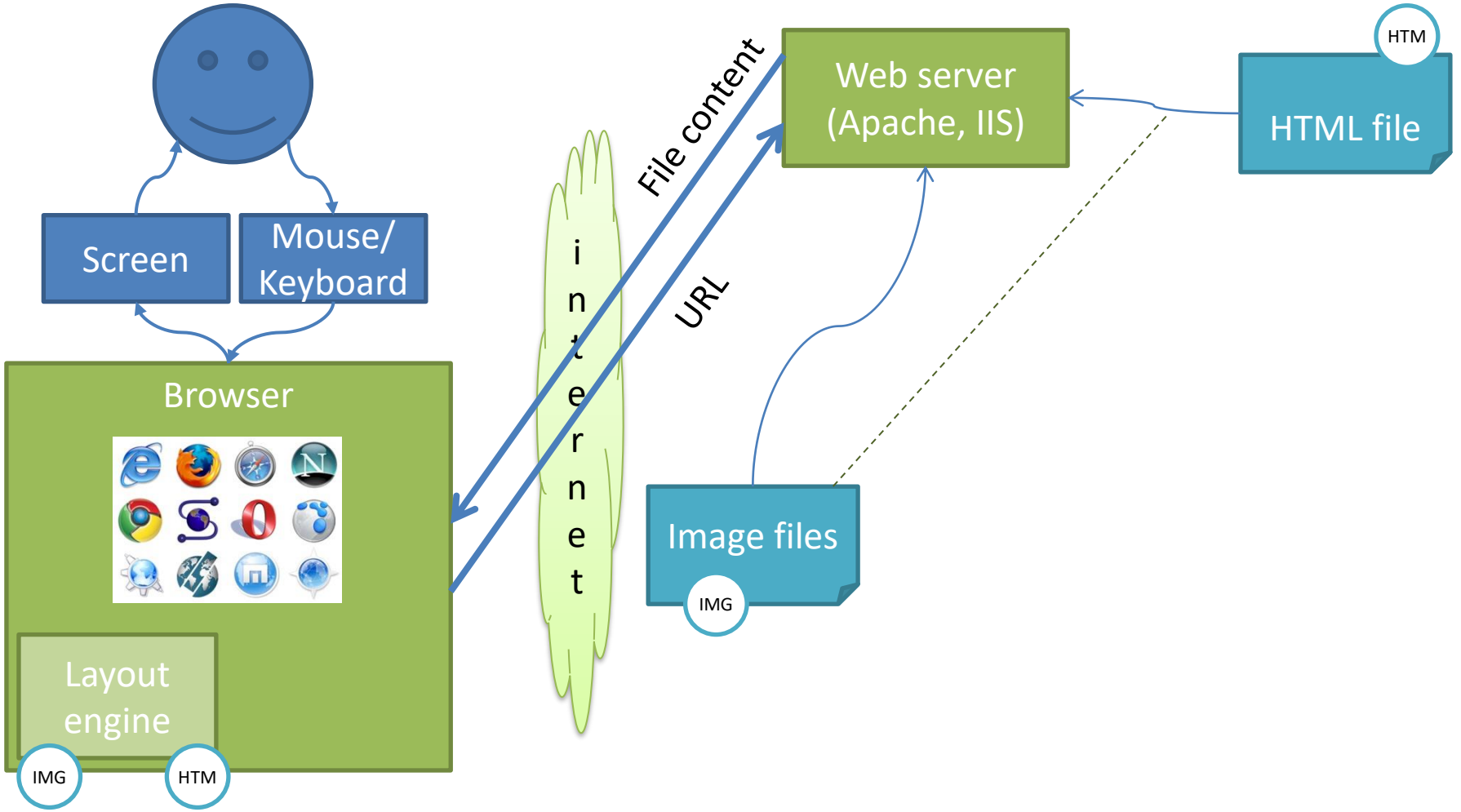
- **Latency**: time required for providing a 0 byte http page. Includes the server activation time, the request decoding time, the file access time, the transmission time and the time for closing the connection.
  - Unit of measure: http/s or s/http
- **Throughput**: maximum speed at which infinite-sized pages can be sent.
  - Unit of measure: Bytes (Mbytes)/s
- #Requests / s
- #Pages / s

Static web transaction

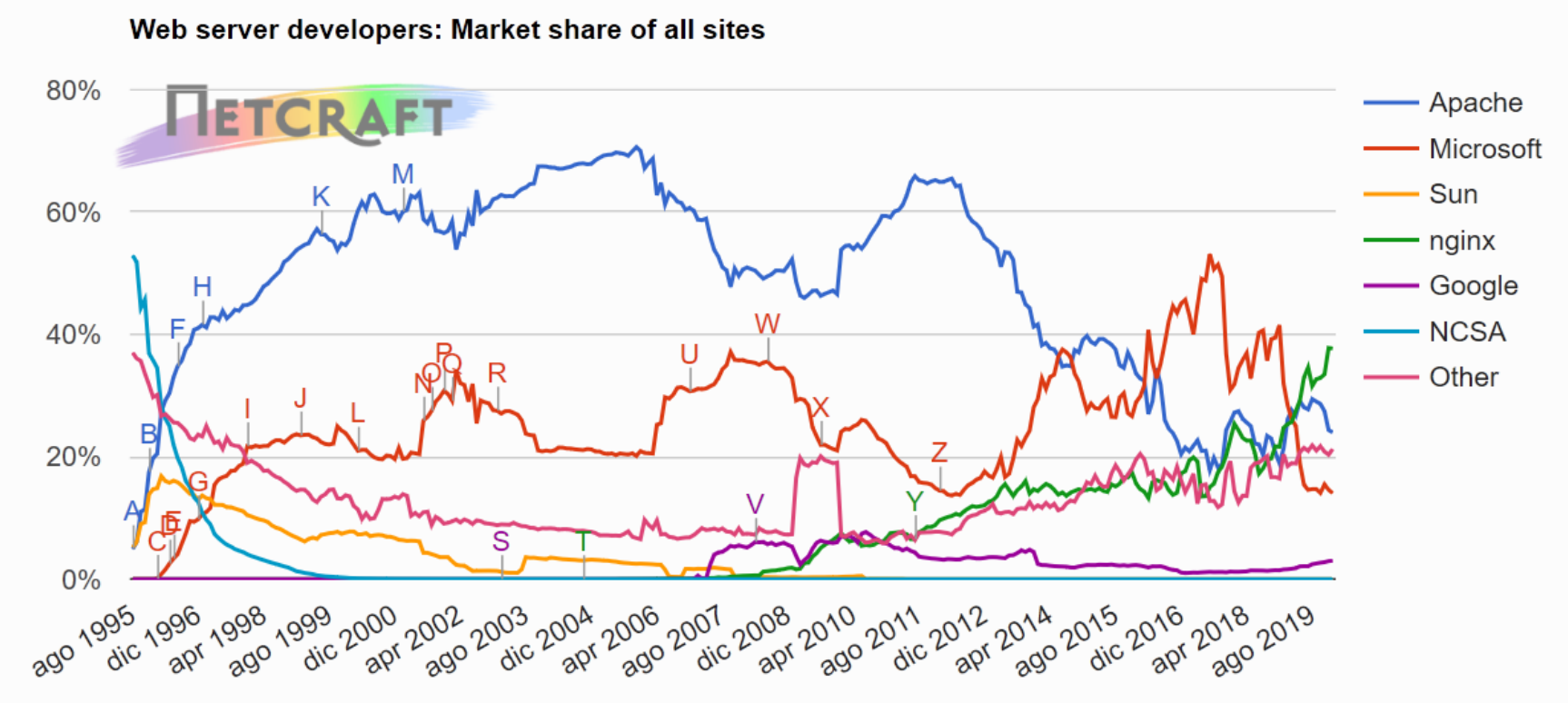
T = Latency + HttpResponseSize / Throughput



General web architecture



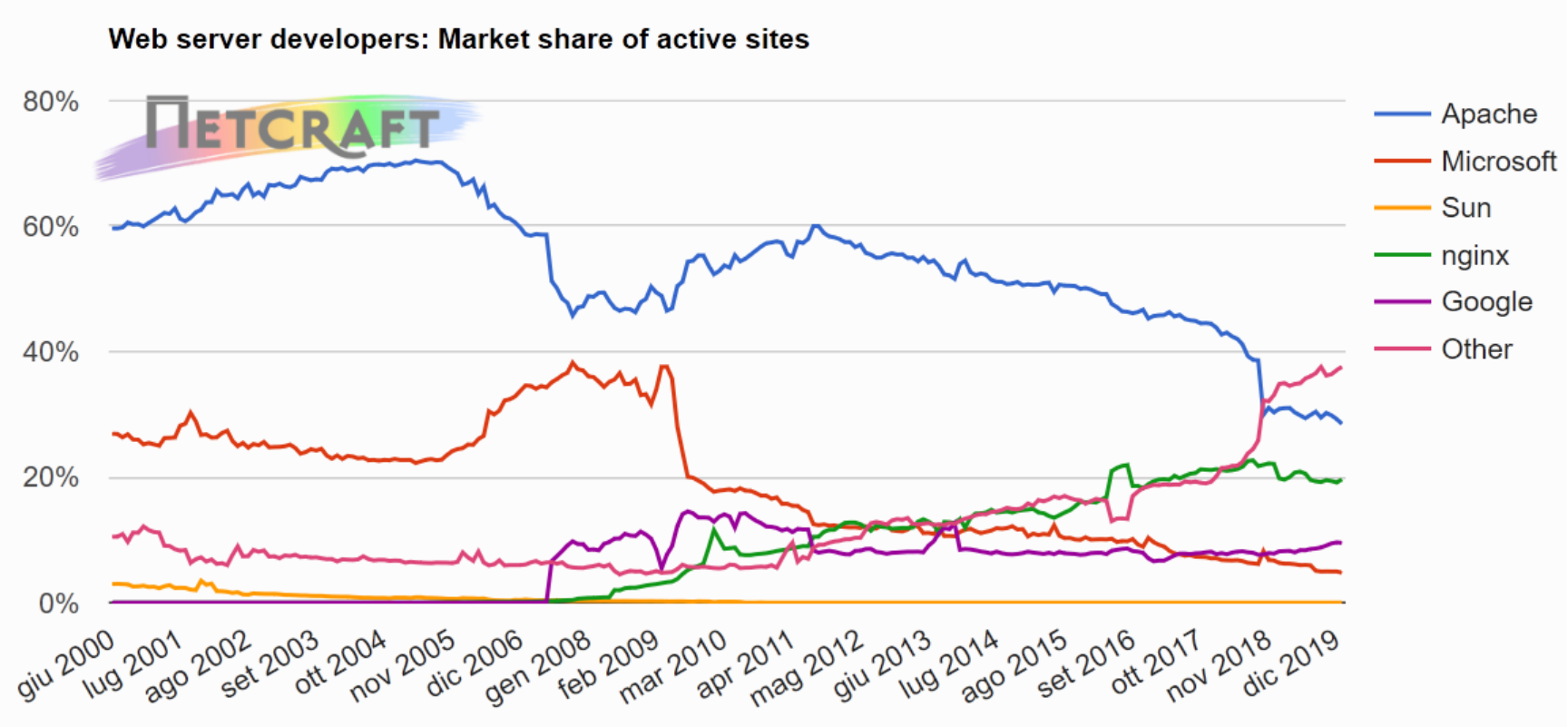
Web Server



Source: http://news.netcraft.com/

https://news.netcraft.com/archives/2020/01/21/january-2020-web-server-survey.html

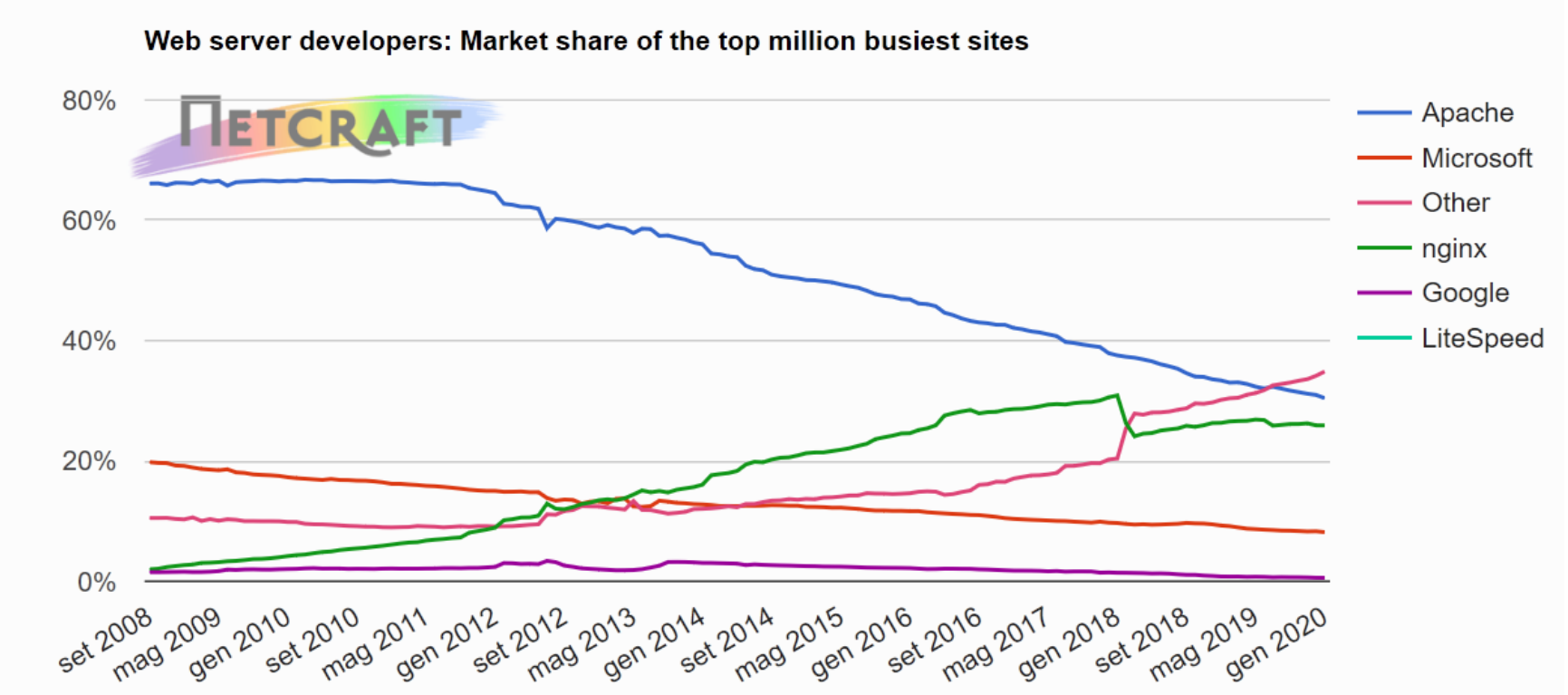
Web Server



Source: http://news.netcraft.com/

https://news.netcraft.com/archives/2020/01/21/january-2020-web-server-survey.html

Web Server



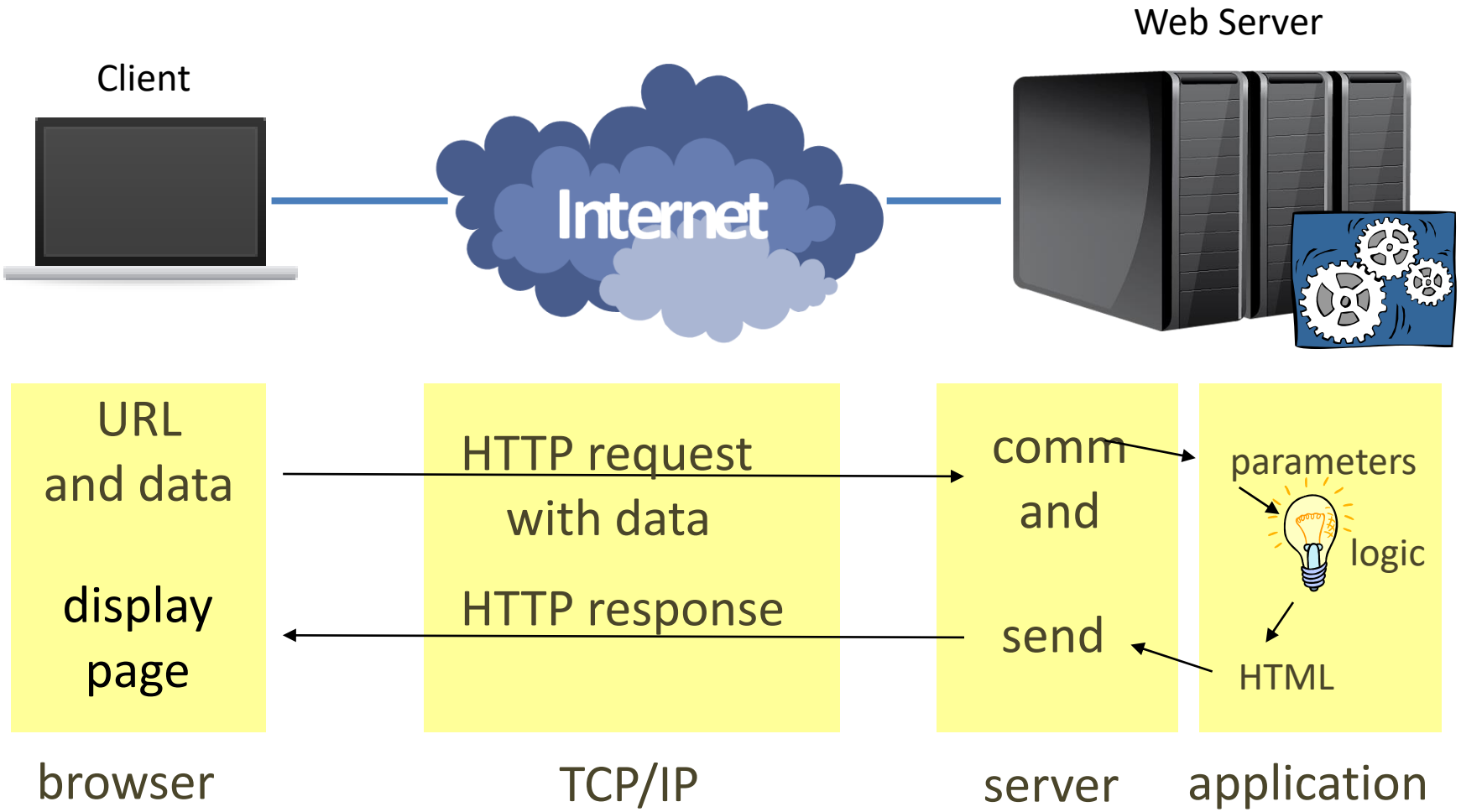
Source: http://news.netcraft.com/

https://news.netcraft.com/archives/2020/01/21/january-2020-web-server-survey.html

Application server

- Dynamic page generation
- Manages the site business logic
- It’s the middle tier between the client browser and the data residing on a database
- Implements the session mechanisms
- Different technologies and architectures are available

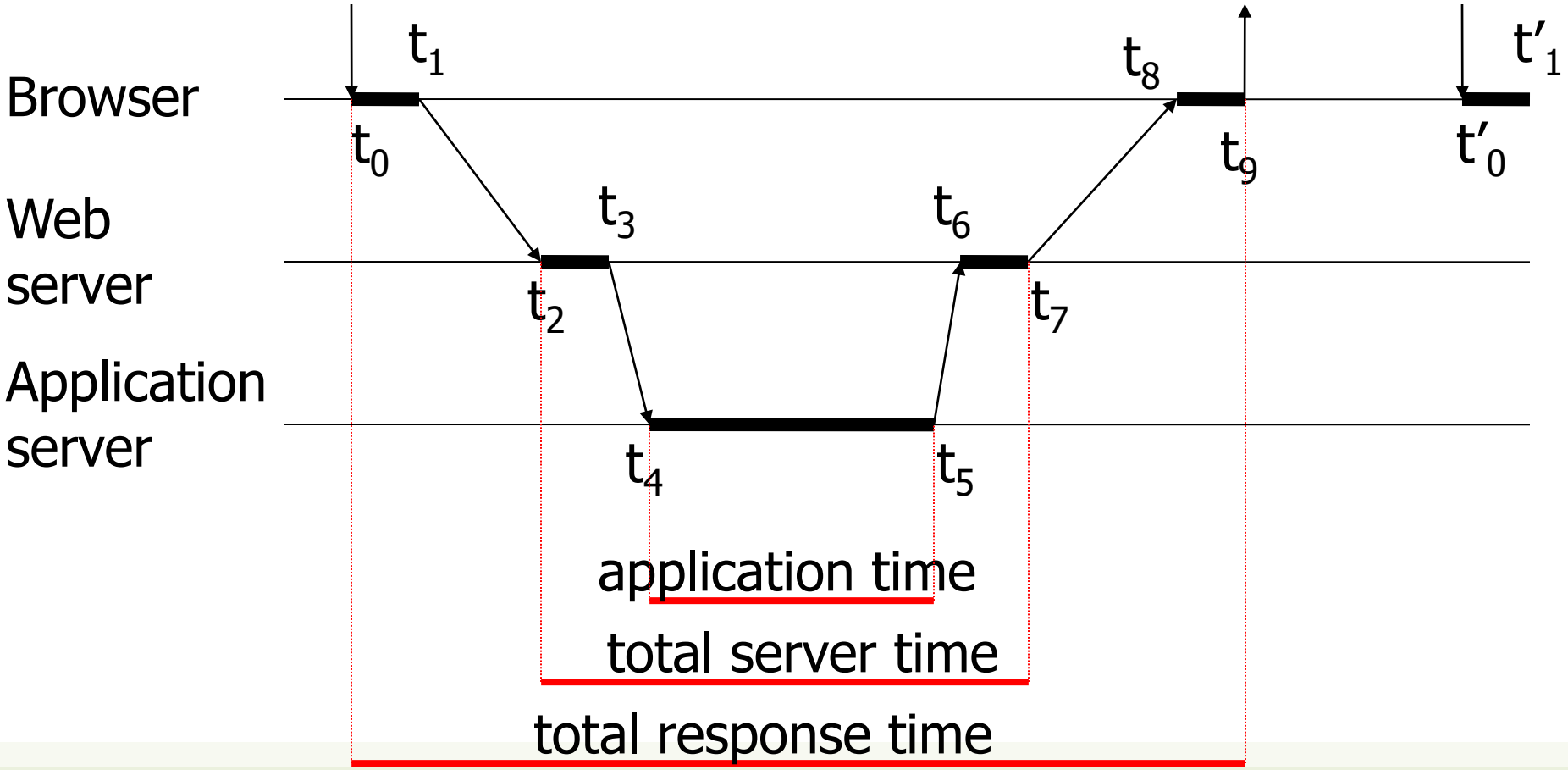
Dynamic web transaction



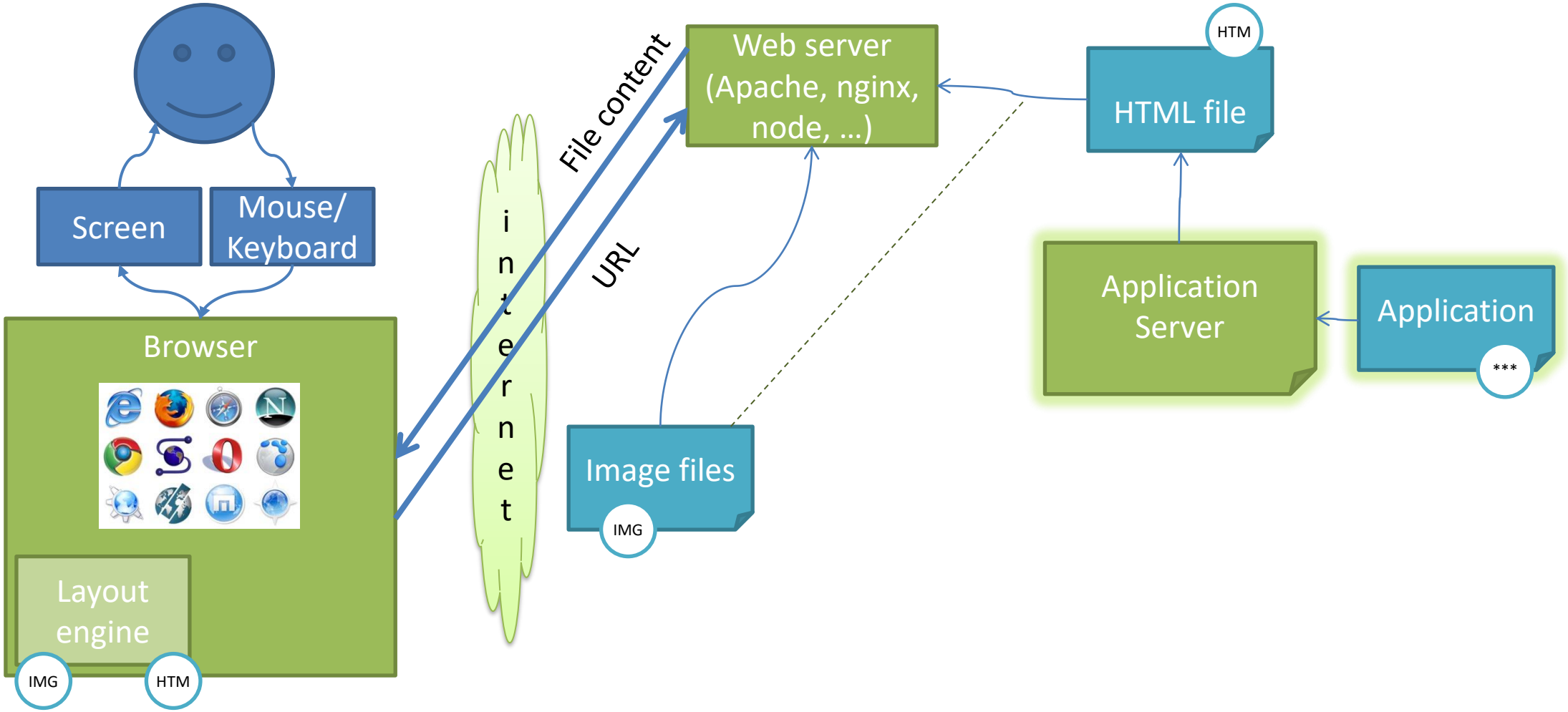
Adopted standards

- HTTP: POST or GET with query, for sending user-specified data
- Integration of a programming language accessible to the Web Server
  - ASP, PHP, PERL, JS, … as new languages for application development
  - Python, Java, C#, … as adaptations of existing languages
- Cookies for storing the state of a session

Dynamic web transaction

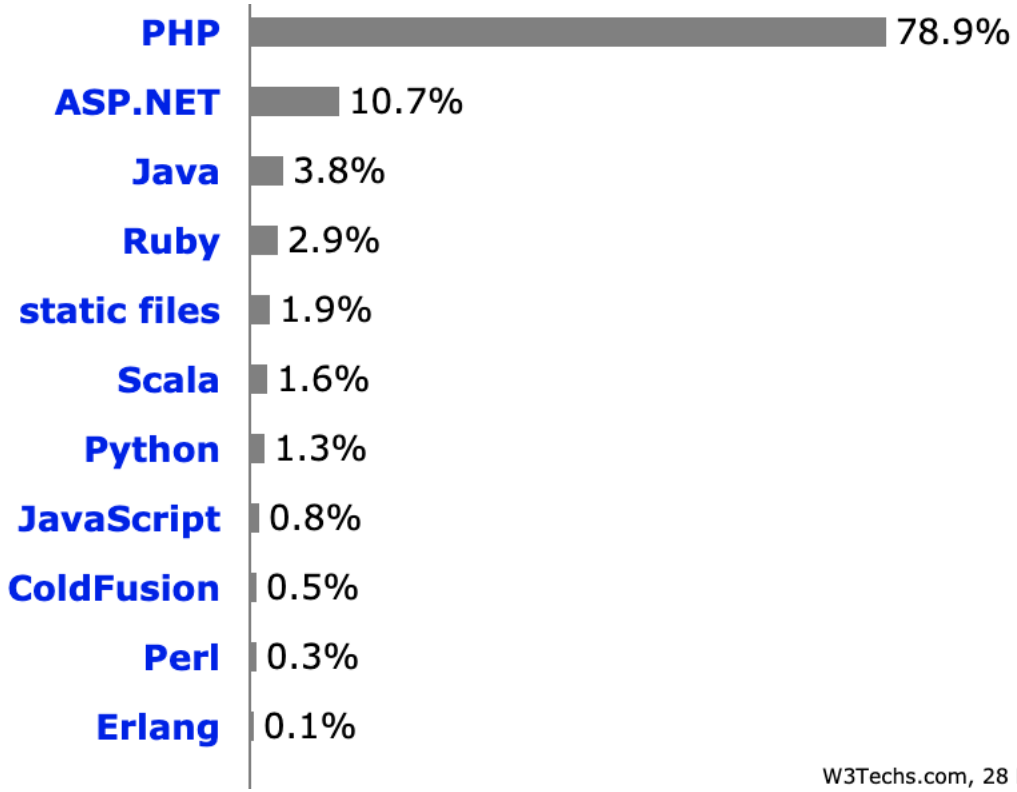


General web architecture



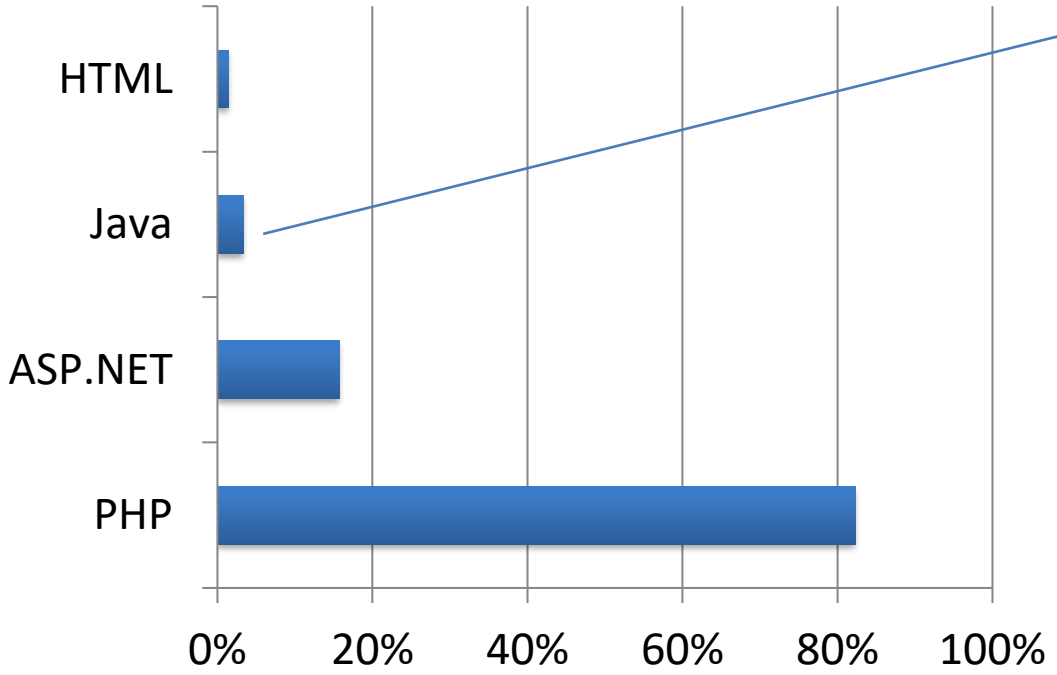
Application Servers

Percentages of websites using various server-side programming languages

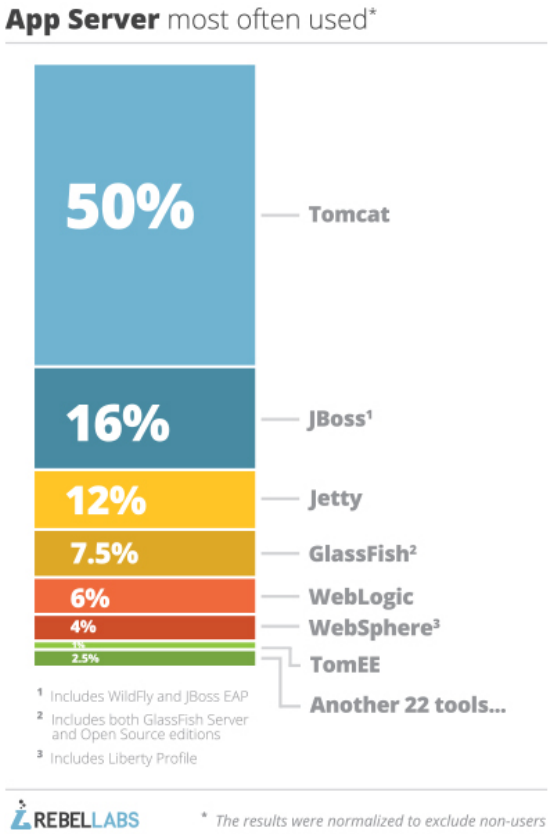


https://w3techs.com/technologies/overview/programming_language/all

Application Servers



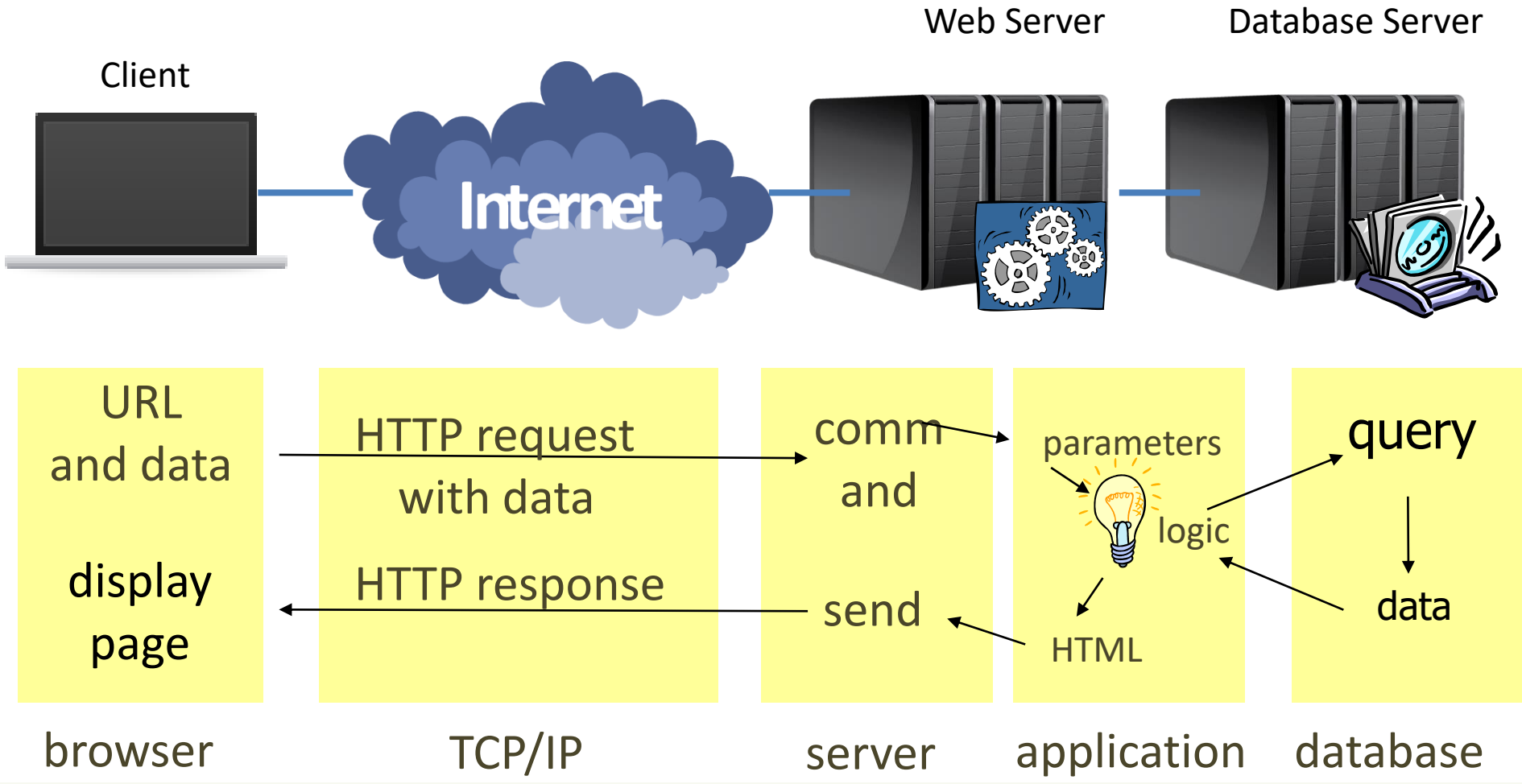
https://en.wikipedia.org/wiki/Website#Dynamic_website



Database server

- Stores the data on which the application server works.
- Executes the queries issued by the application server:
  - Updates the stored data
  - Inserts new data
  - Provides back query results
- The most frequent/complex queries can be implemented internally as stored procedures (pre-compiled queries with parameters)

DB-backed web transaction



Adopted standards

- SQL (structured query language)

Database server

- Queries are almost always in SQL
  - SELECT * FROM table;
  - ….
- Often adopts the relational database model
  - Other models can be used
    - Object model
    - Triple model
- The most advanced/complete solutions are called Transaction servers

Example (PHP)

The application composes the query

```
<?php
$query = “SELECT doc_id FROM key_doc_index, keywords WHERE
key_doc_index.key_id = keywords.id AND keywords.key =
$_REQUEST[“query”];”;
```

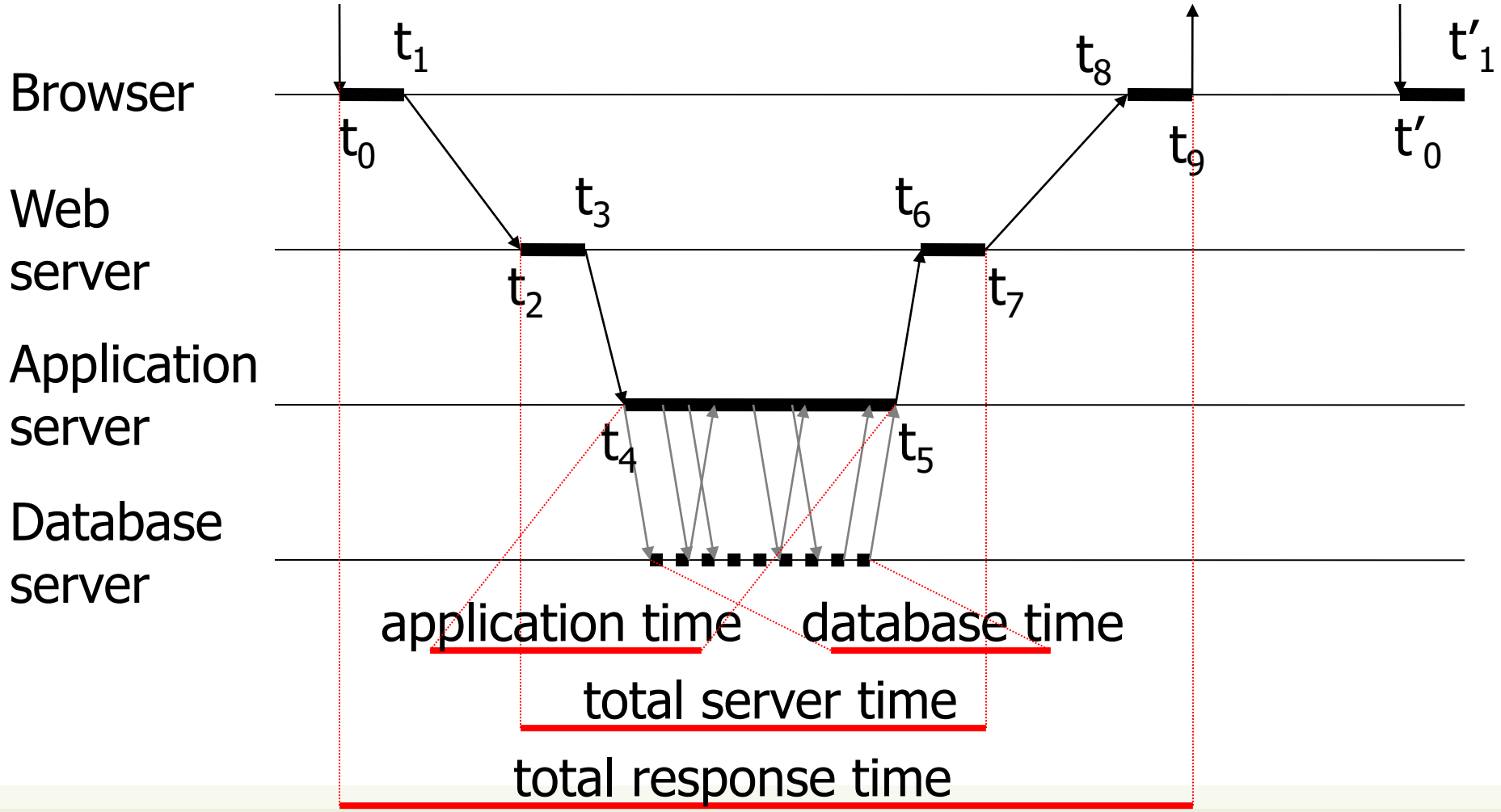
The query is sent to the db-server and a rowset containing the results is returned

```
$rowset = mysql_query($query);
```

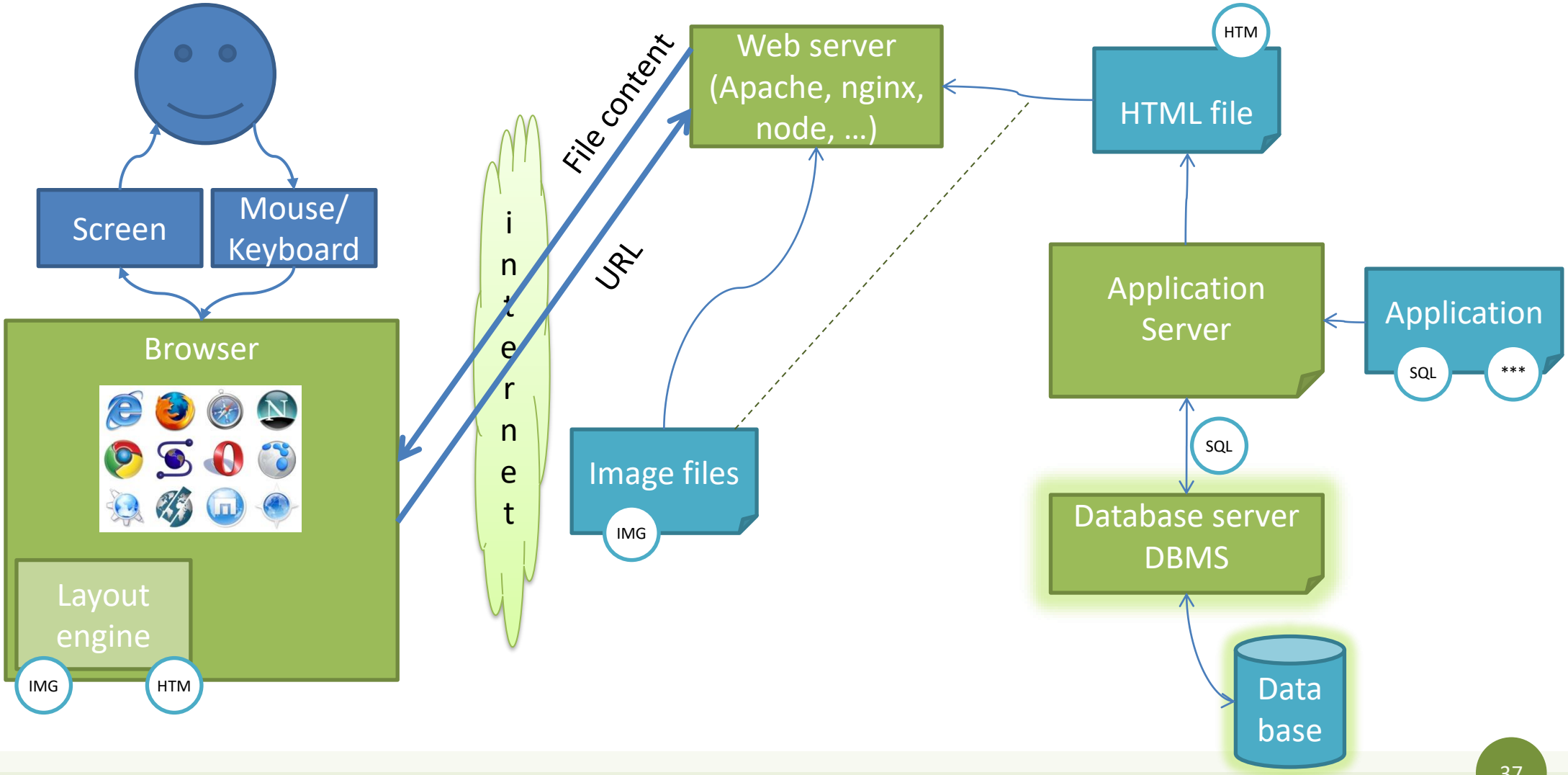
```
while($row = mysql_fetch_row($rowset))
{
//elaborate data
}
?>
```

The application elaborates the data

Database-driven transaction



General web architecture



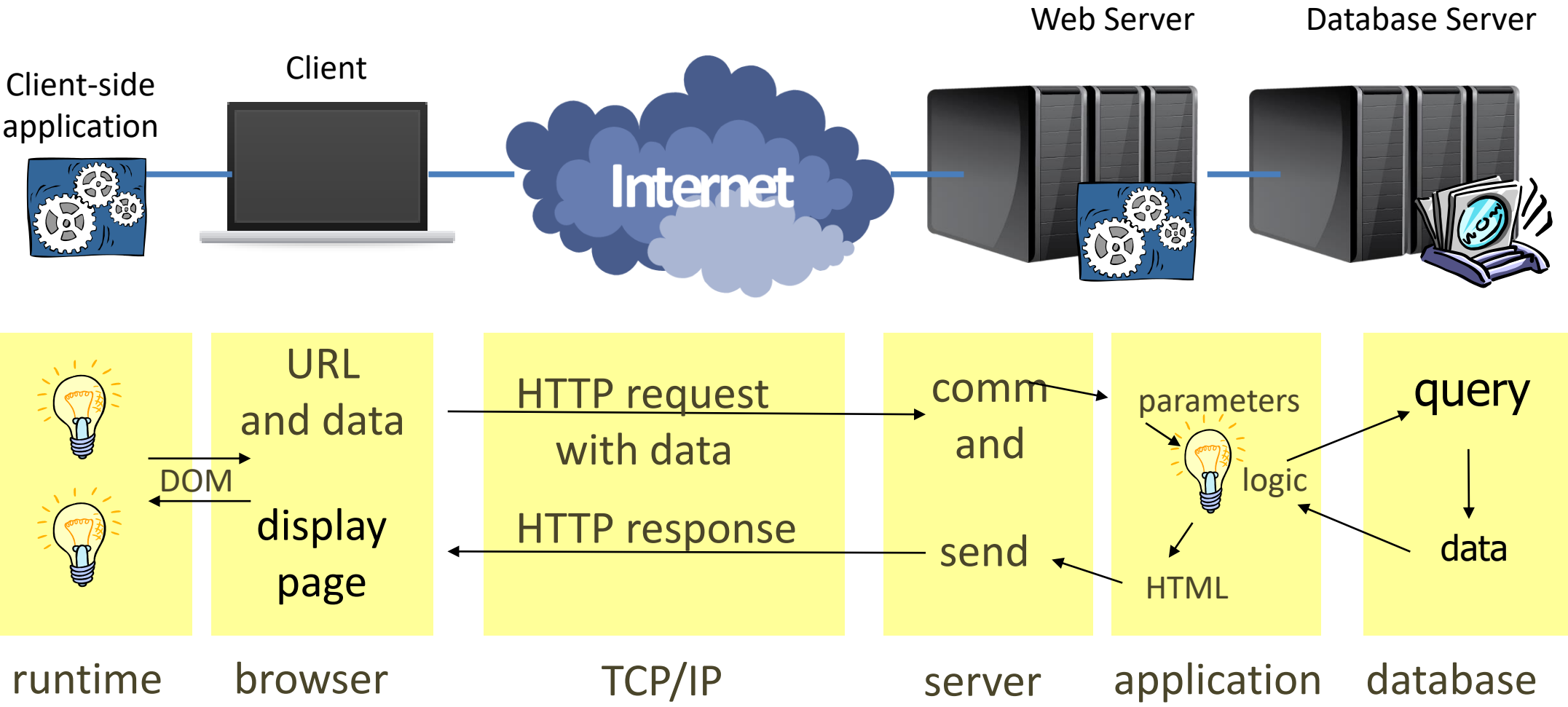
Client-side programming

- Making a web page dynamic
  - Able to change the page content after is was loaded by the server
  - Able to interact with the user, on the browser
  - Able to “augment” the default interactions offered by the browser
- Examples:
  - Animations on the page
    - e.g. menus, accordions, slideshows, hide/show, …
  - Form validation

Client-side programming

- Requires:
  - A programming language accepted by all browsers
  - A program embedded in the web page
  - An execution engine in the browser
- Limitations:
  - All data needed by the program must be known beforehand (when the page is loaded)
  - The program must have a restricted access to the execution environment

Rich-client web transaction

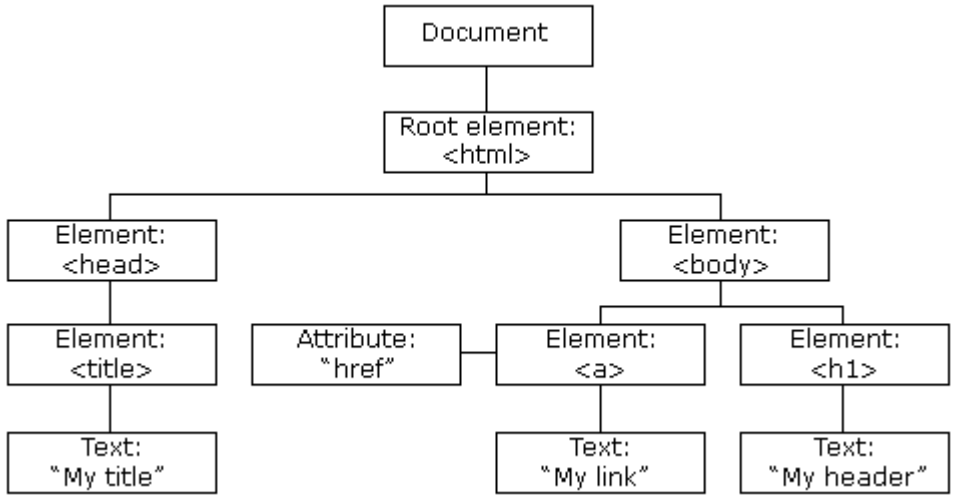


Document Object Model (DOM)

"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

- Standard data structure for representing the web page content
- Supported by all browsers
- Javascript programs can read & modify the DOM
- Abstracts
  - Browser
  - HTML

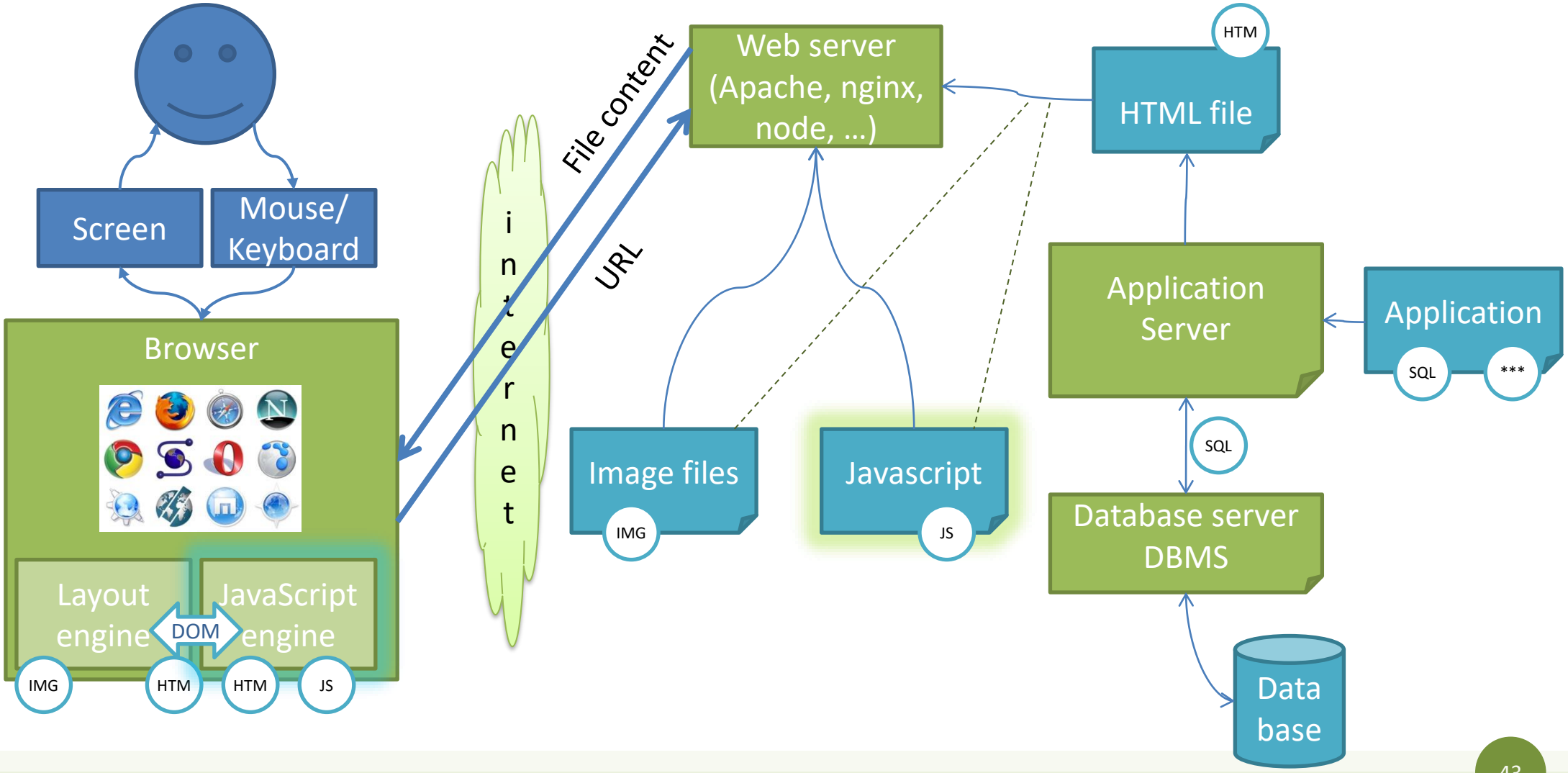
- The HTML DOM is a standard for how to get, change, add, or delete HTML elements



Adopted standards

- DOM, JavaScript, CSS
  - JavaScript to handle a runtime environment on the browser
  - DOM (Document Object Model) to allow access and on-the fly modification of the web page
  - CSS to modify attributes and handle objects

General web architecture

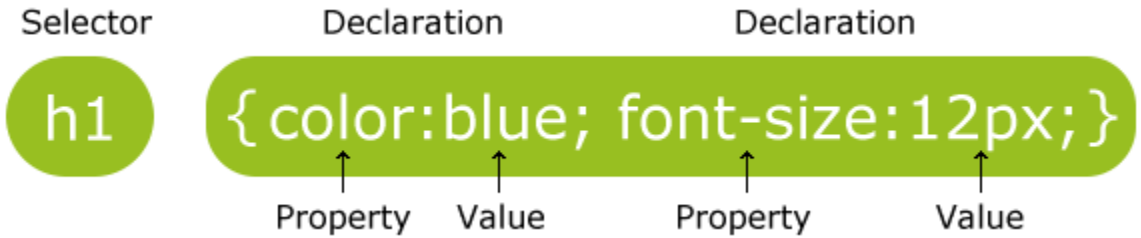


Separating layout from content

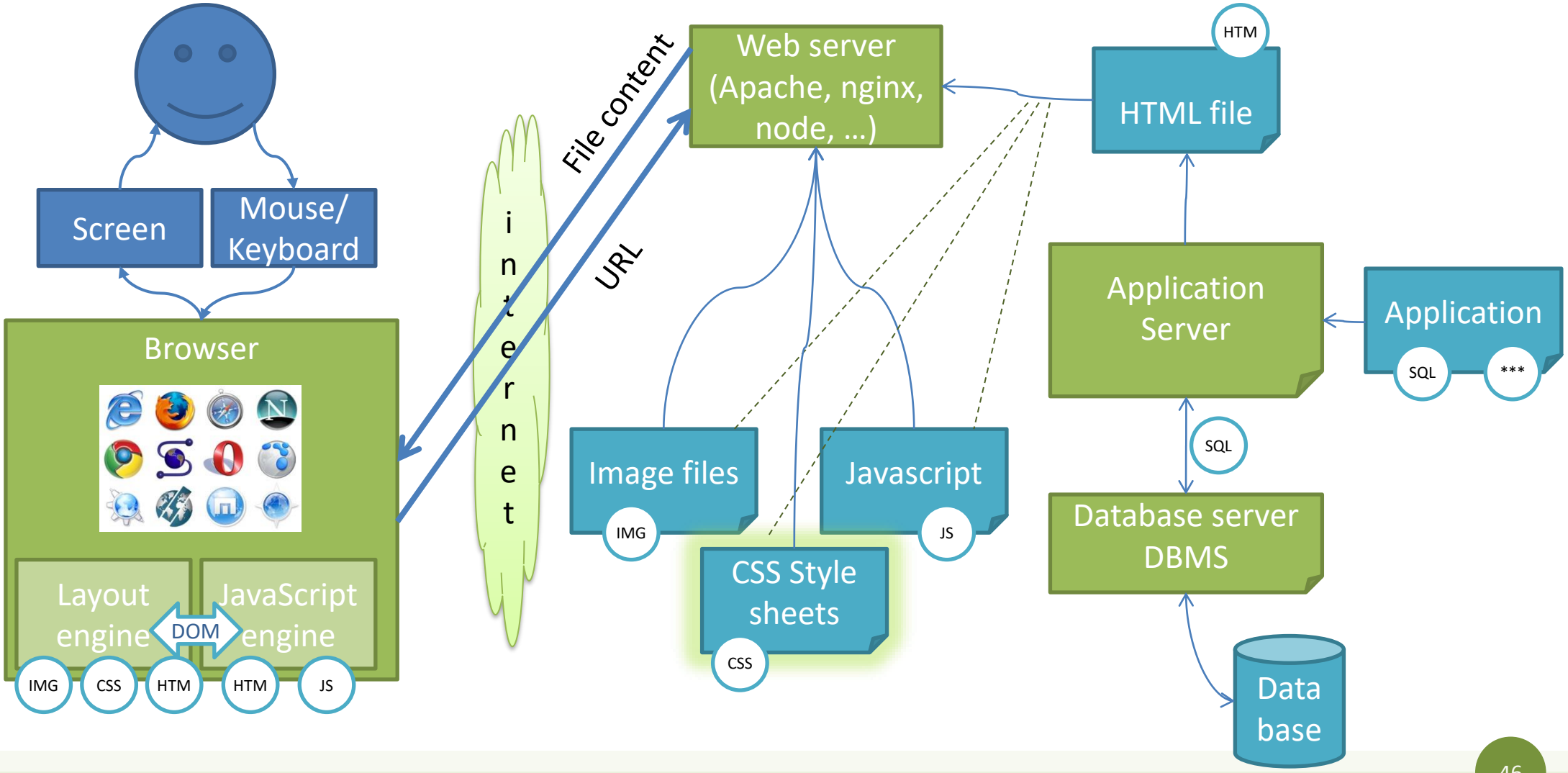
- Goals:
  - Allow the definition of complex layouts
  - Adapt web pages to different resolutions
  - Adapt web pages to different devices (e.g., mobile)
  - Adapt web pages to different preferences (e.g., color schemes)
  - Adapt web pages to different media (e.g., text vs video)
  - In a standard way ☺
- 'Adapt' means:
  - Resize, Reflow, Show/Hide, Substitute, Animate, Highlight, Move, …
- Solution: Cascading Style Sheets (CSS)

CSS

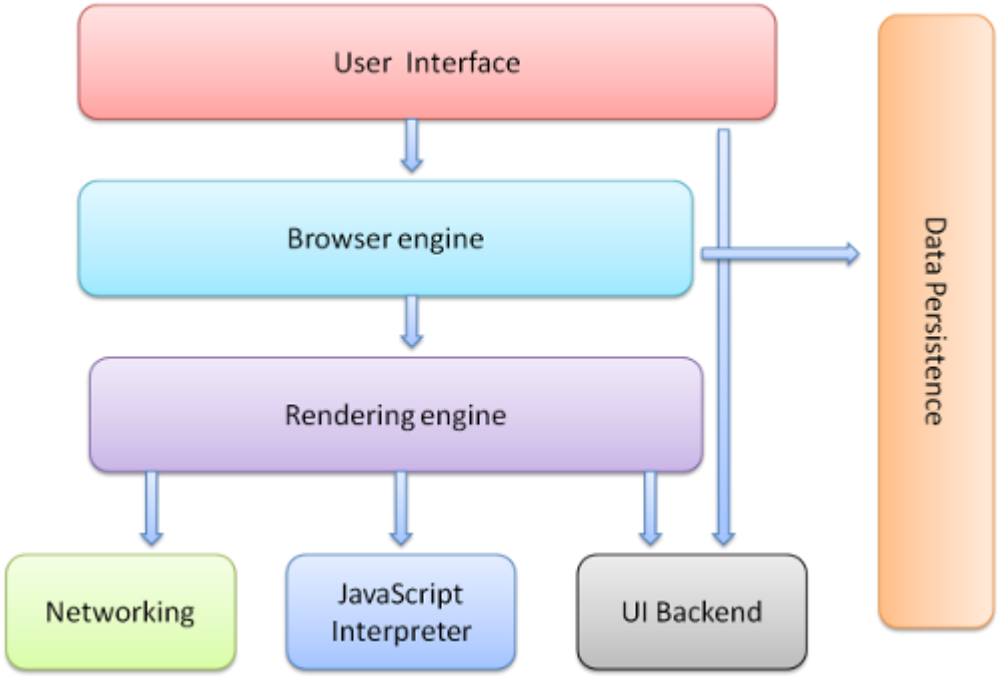
- A set of “Declarations” applied to some “Selector”
  - Selectors identify portions of the web page DOM
  - Declarations set the value of some “properties”
  - Properties control everything: color, size, font, alignment, border, shadow, position, selection status, transitions, links, buttons, cursors, …



General web architecture

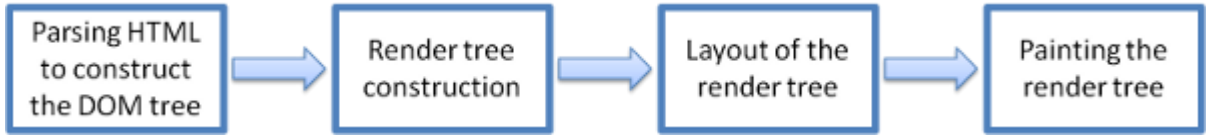


Conceptual Browser architecture

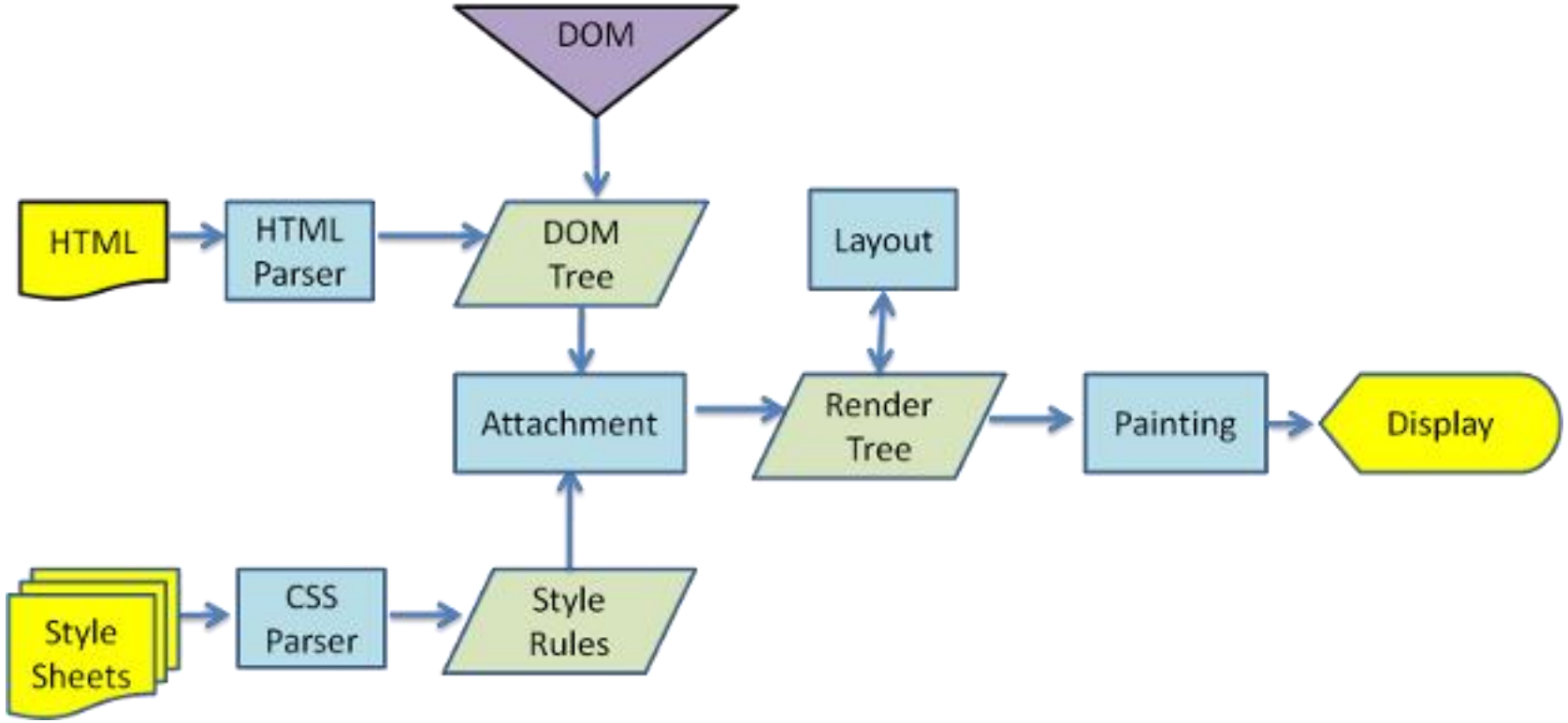


- **The user interface**: this includes the address bar, back/forward button, bookmarking menu, etc. Every part of the browser display except the window where you see the requested page.
- **The browser engine**: marshals actions between the UI and the rendering engine.
- **The rendering engine** : responsible for displaying requested content. For example if the requested content is HTML, the rendering engine parses HTML and CSS, and displays the parsed content on the screen.
- **Networking**: for network calls such as HTTP requests, using different implementations for different platform behind a platform-independent interface.
- **UI backend**: used for drawing basic widgets like combo boxes and windows. This backend exposes a generic interface that is not platform specific. Underneath it uses operating system user interface methods.
- **JavaScript interpreter**. Used to parse and execute JavaScript code.
- **Data storage**. This is a persistence layer. The browser may need to save all sorts of data locally, such as cookies. Browsers also support storage mechanisms such as localStorage, IndexedDB, WebSQL and FileSystem.

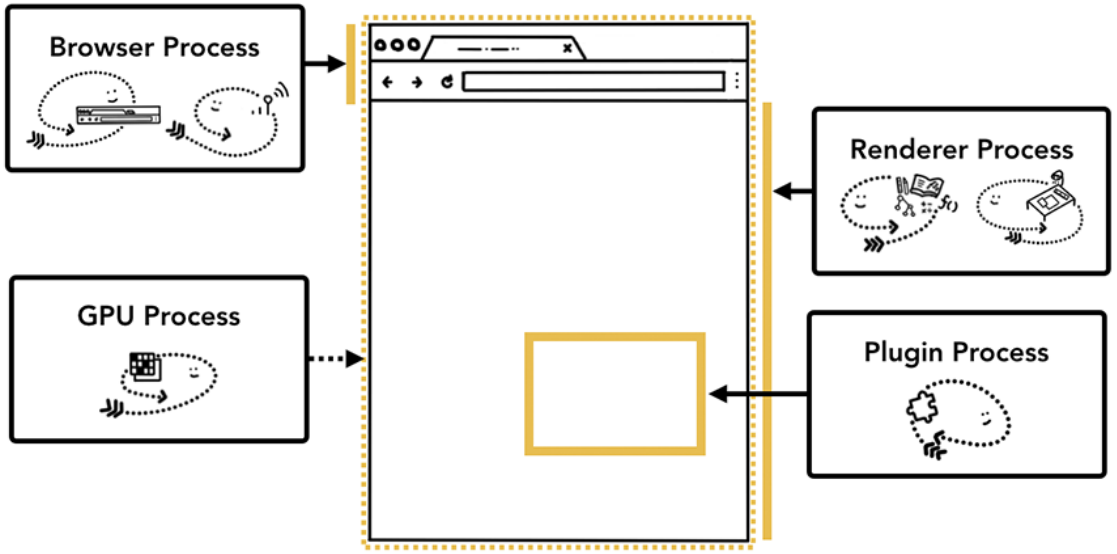
Rendering Engine



Rendering process in WebKit



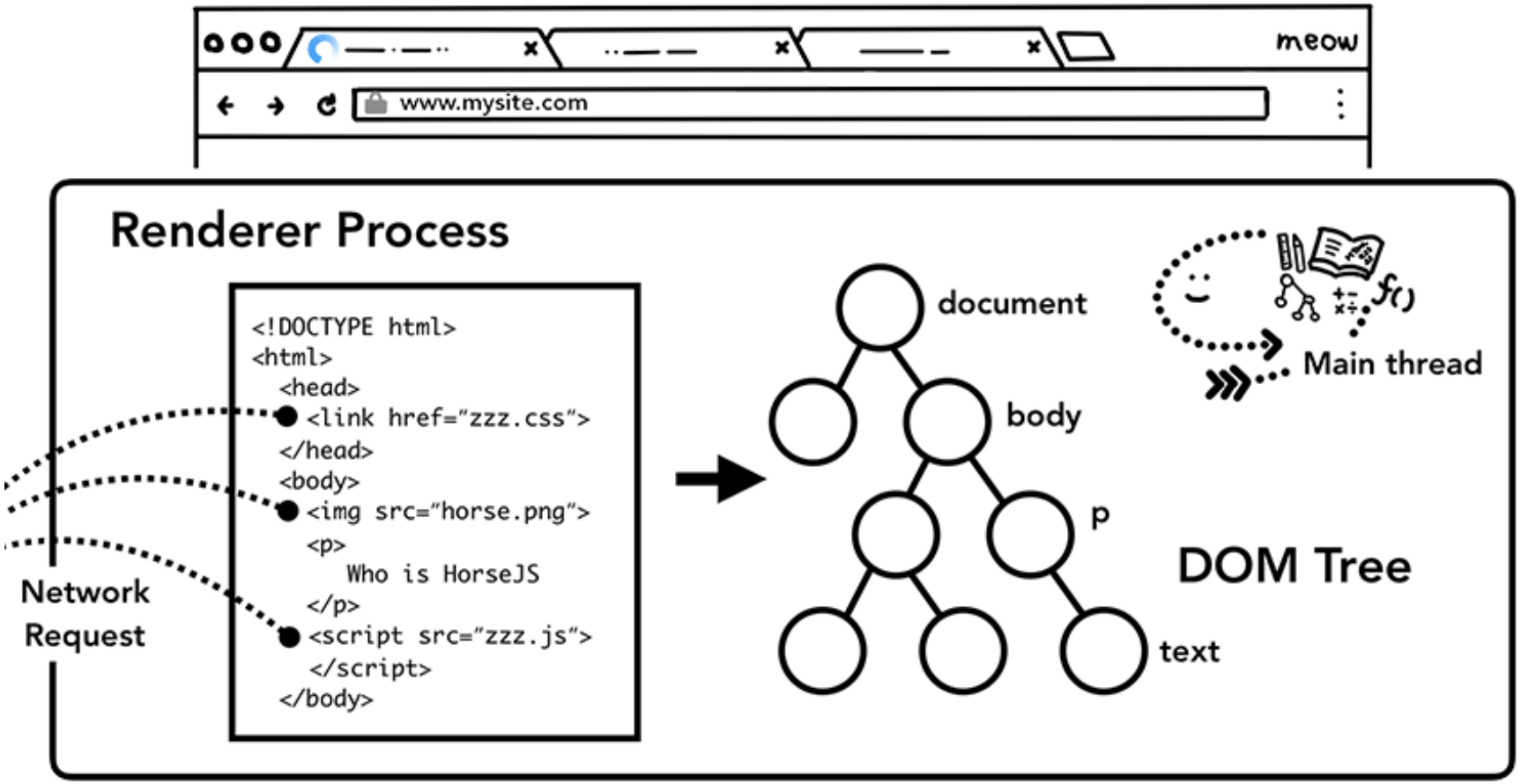
Chrome’s multi-process architecture



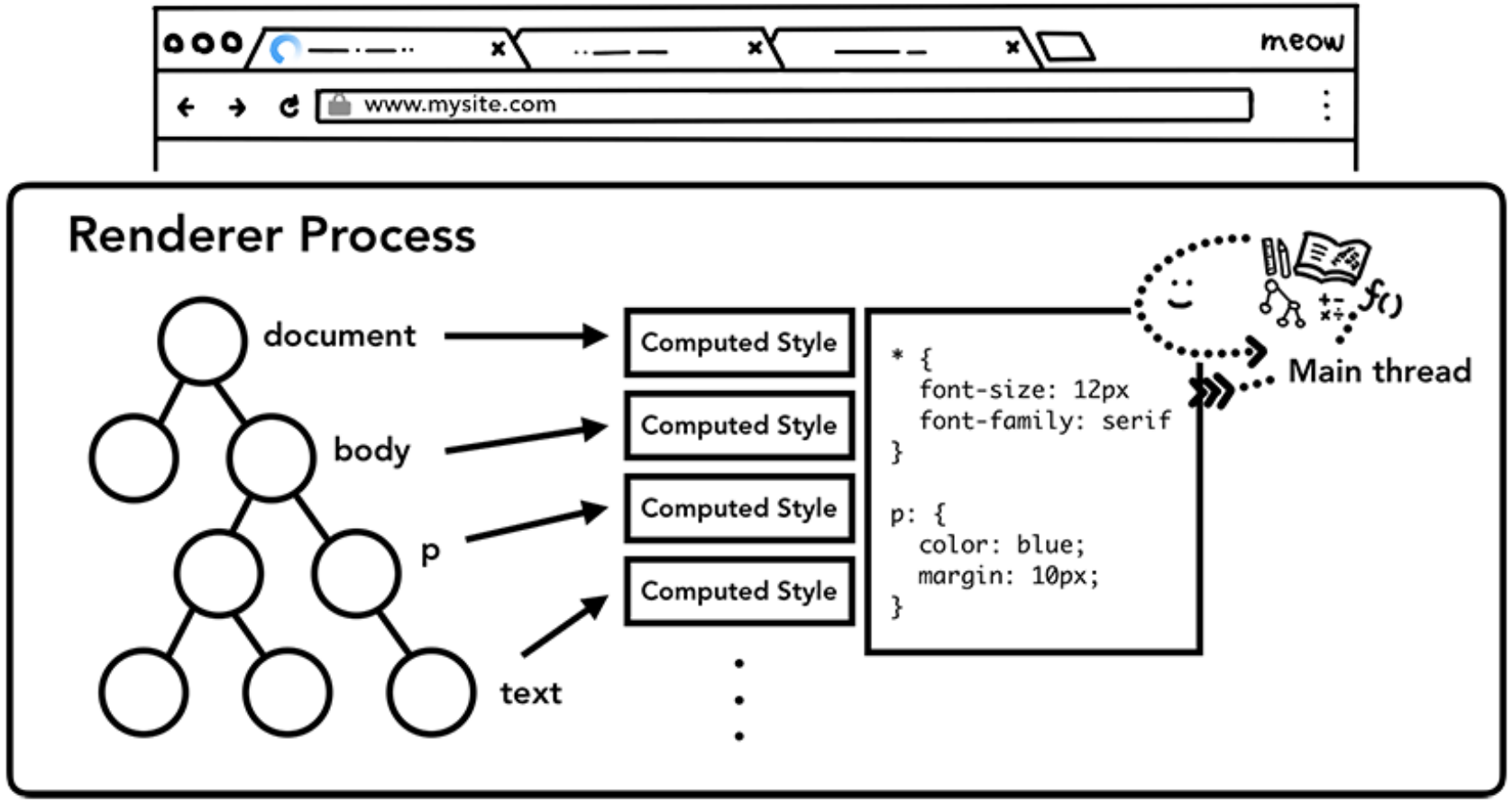
Browser	Controls "chrome" part of the application including address bar, bookmarks, back and forward buttons. Also handles the invisible, privileged parts of a web browser such as network requests and file access.
Renderer	Controls anything inside of the tab where a website is displayed.
Plugin	Controls any plugins used by the website, for example, flash.
GPU	Handles GPU tasks in isolation from other processes. It is separated into different process because GPUs handles requests from multiple apps and draw them in the same surface.

Source: Inside look at modern web browser (Google)

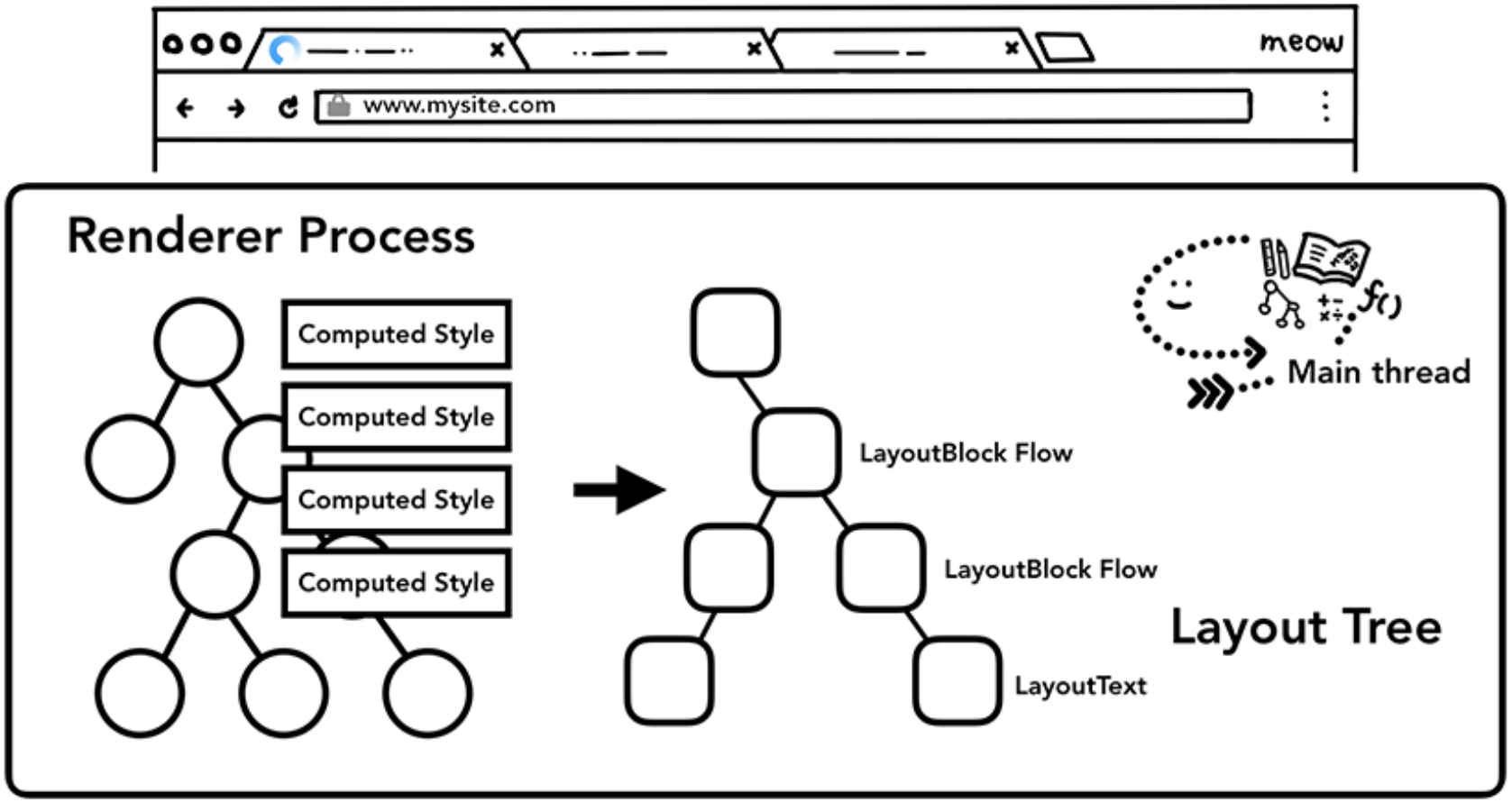
Parsing HTML and building a DOM tree



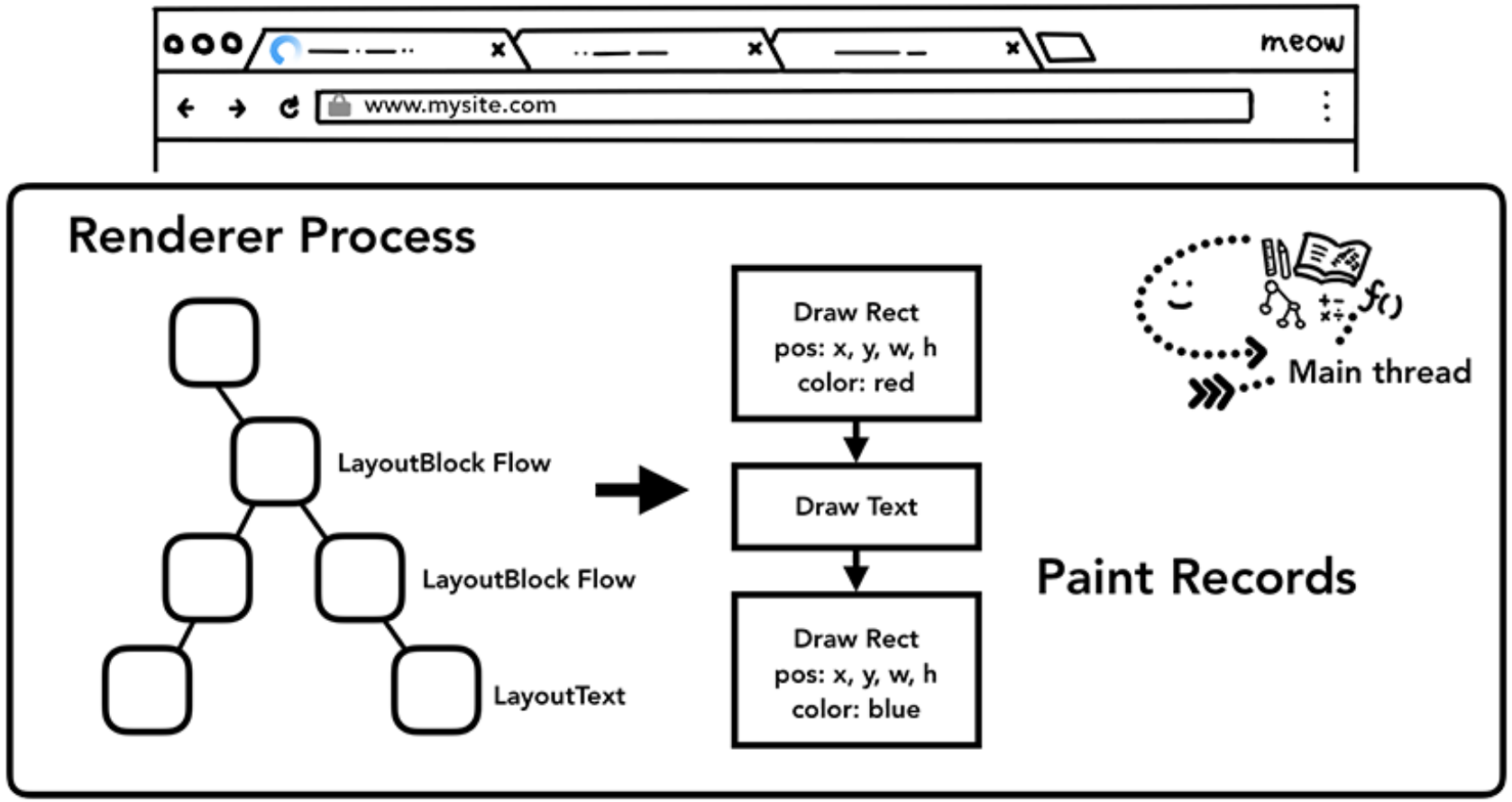
Parsing CSS: add computed style to each node



Produce Layout Tree from DOM+Computed Styles

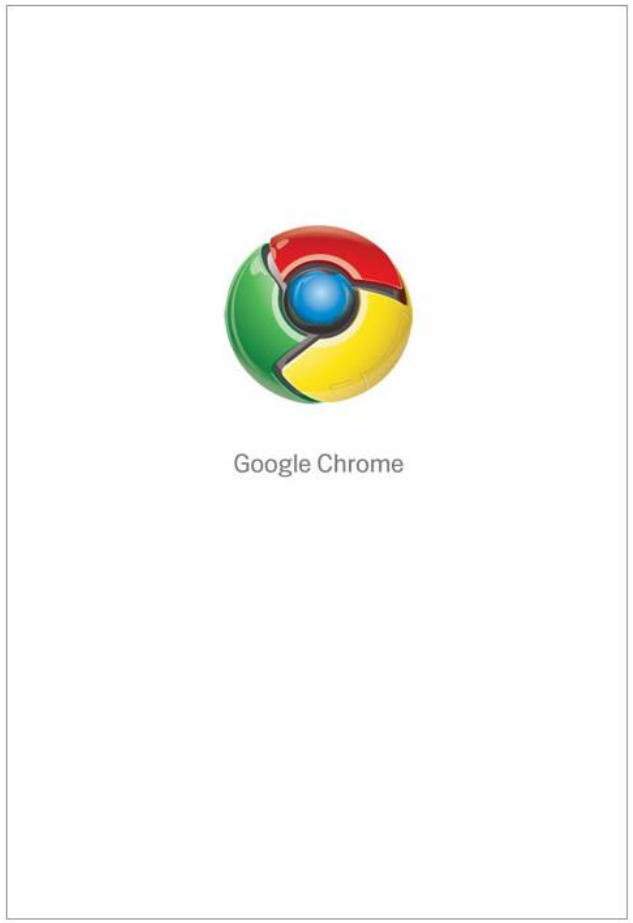


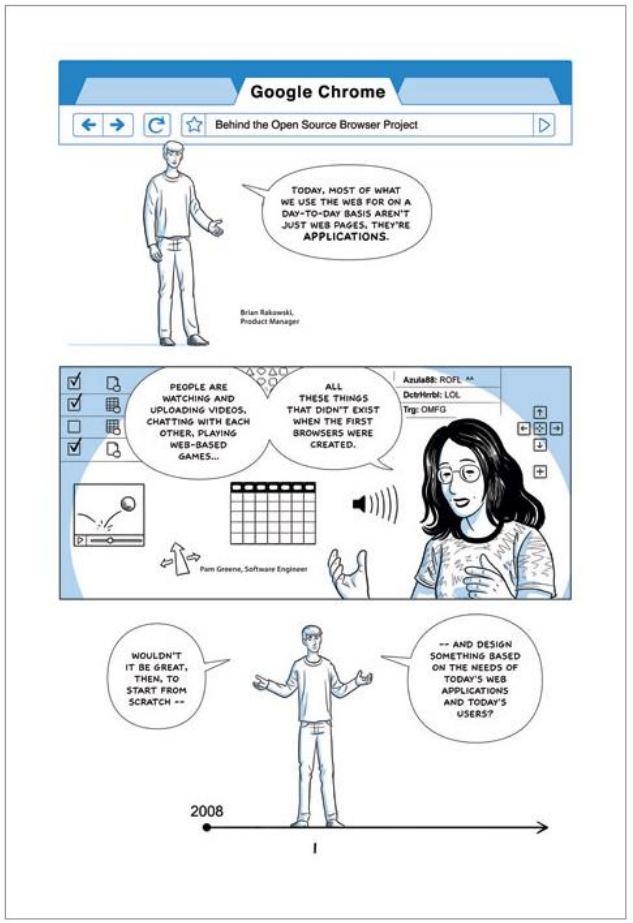
From Layout decisions to Paint actions



Suggested reading

https://www.google.com/googlebooks/chrome/

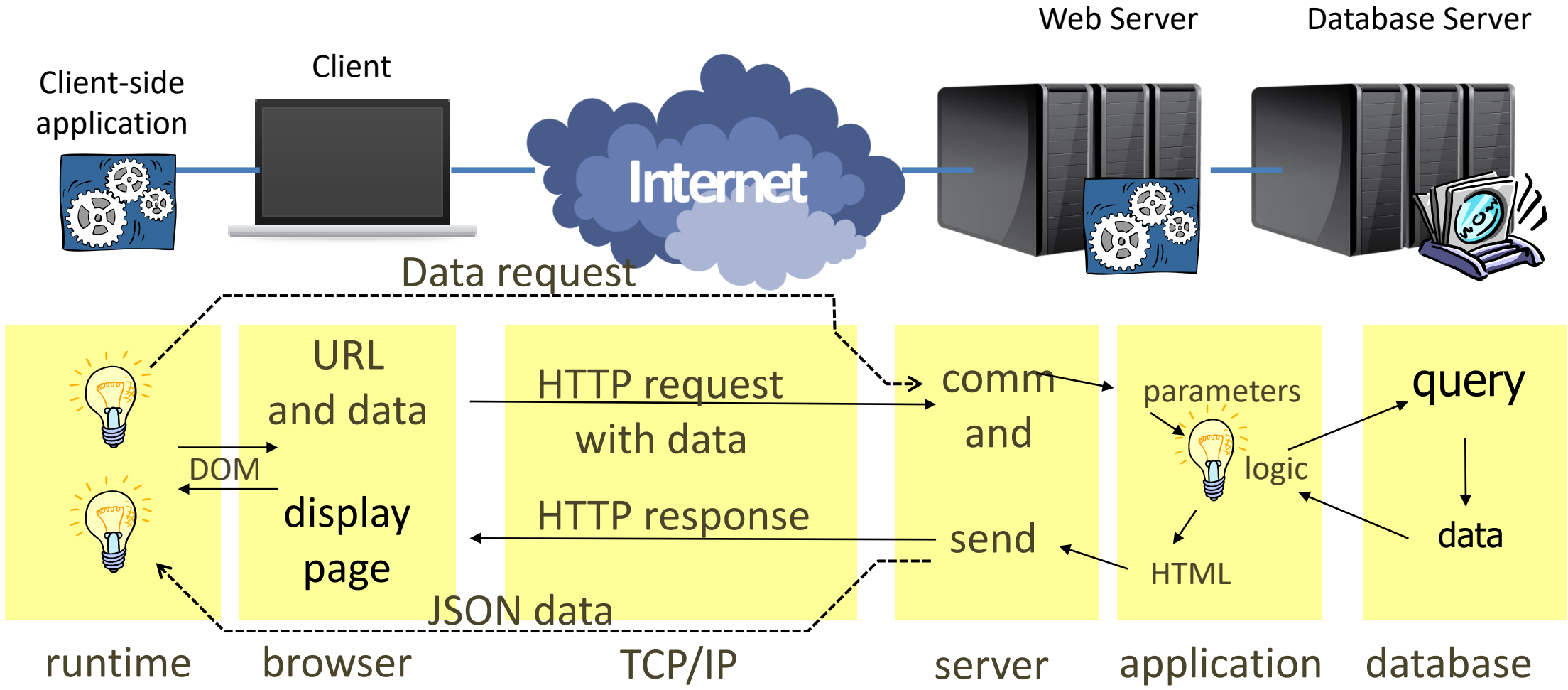




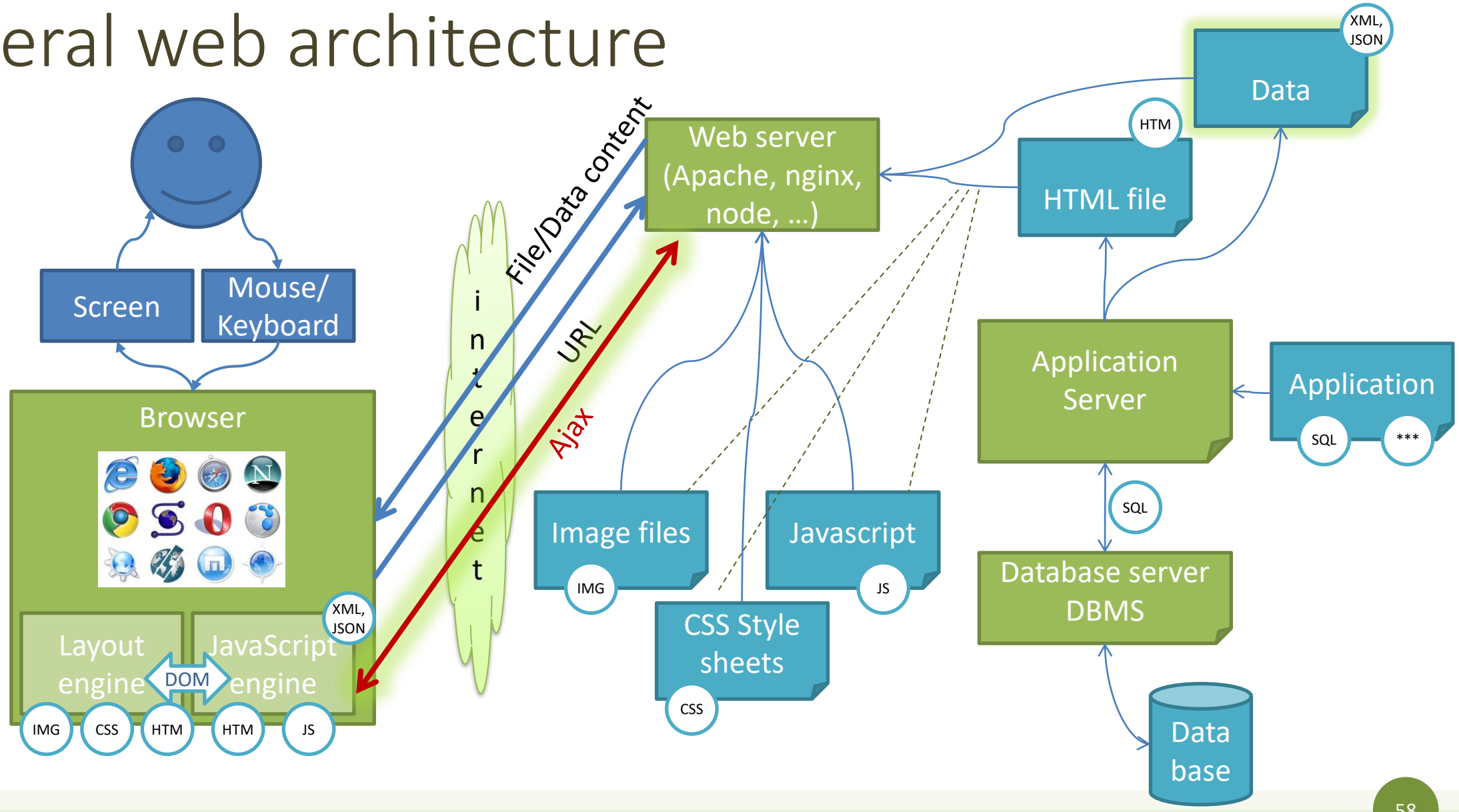
Web 2.0

- Web applications support social interaction models
- Rich, dynamic, interactive user interfaces
- Continuous update
  - Client and Server “chatting”
- Integration of contents across web sites (mashups, cloud services)

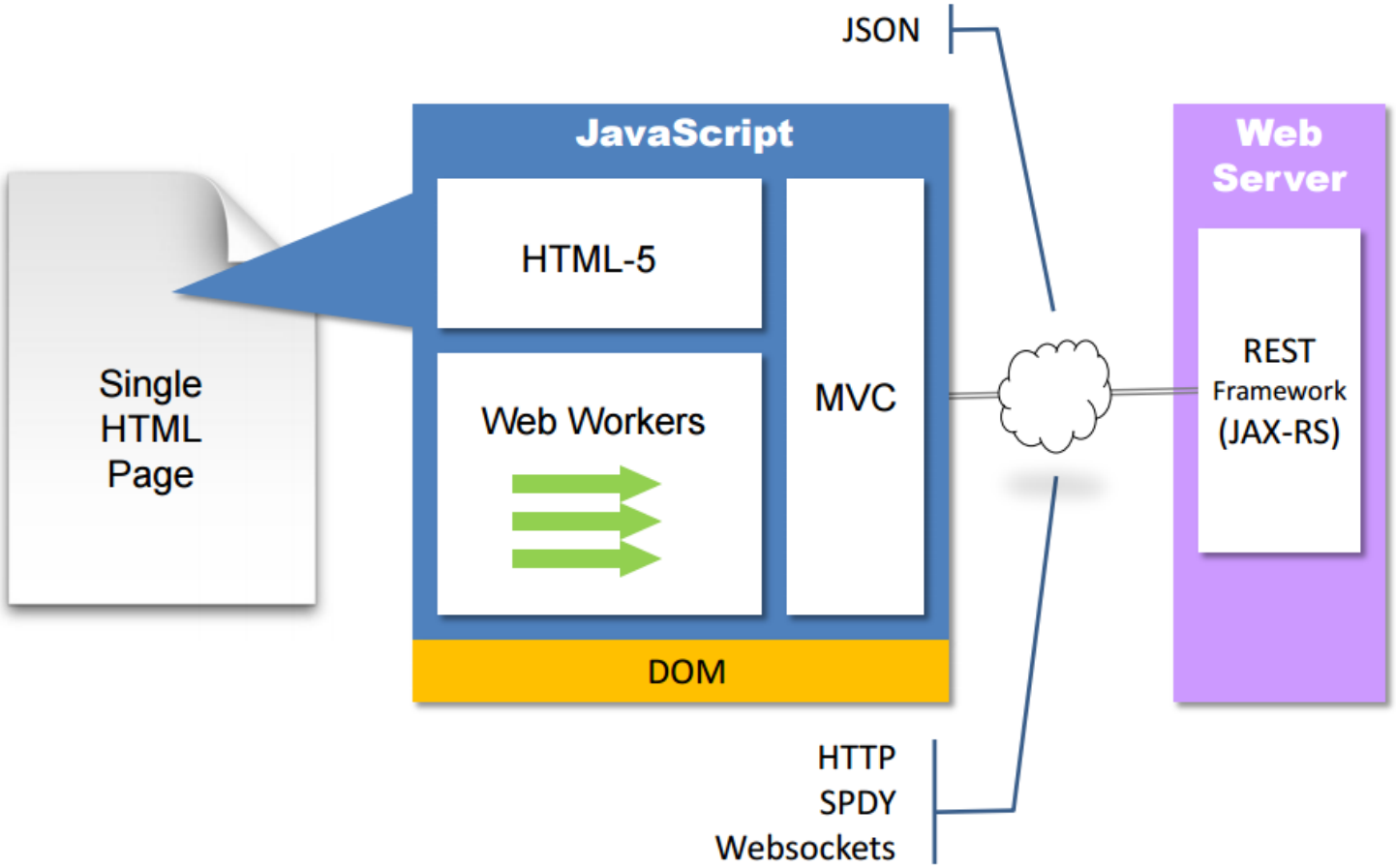
Asynchronous Ajax requests



General web architecture



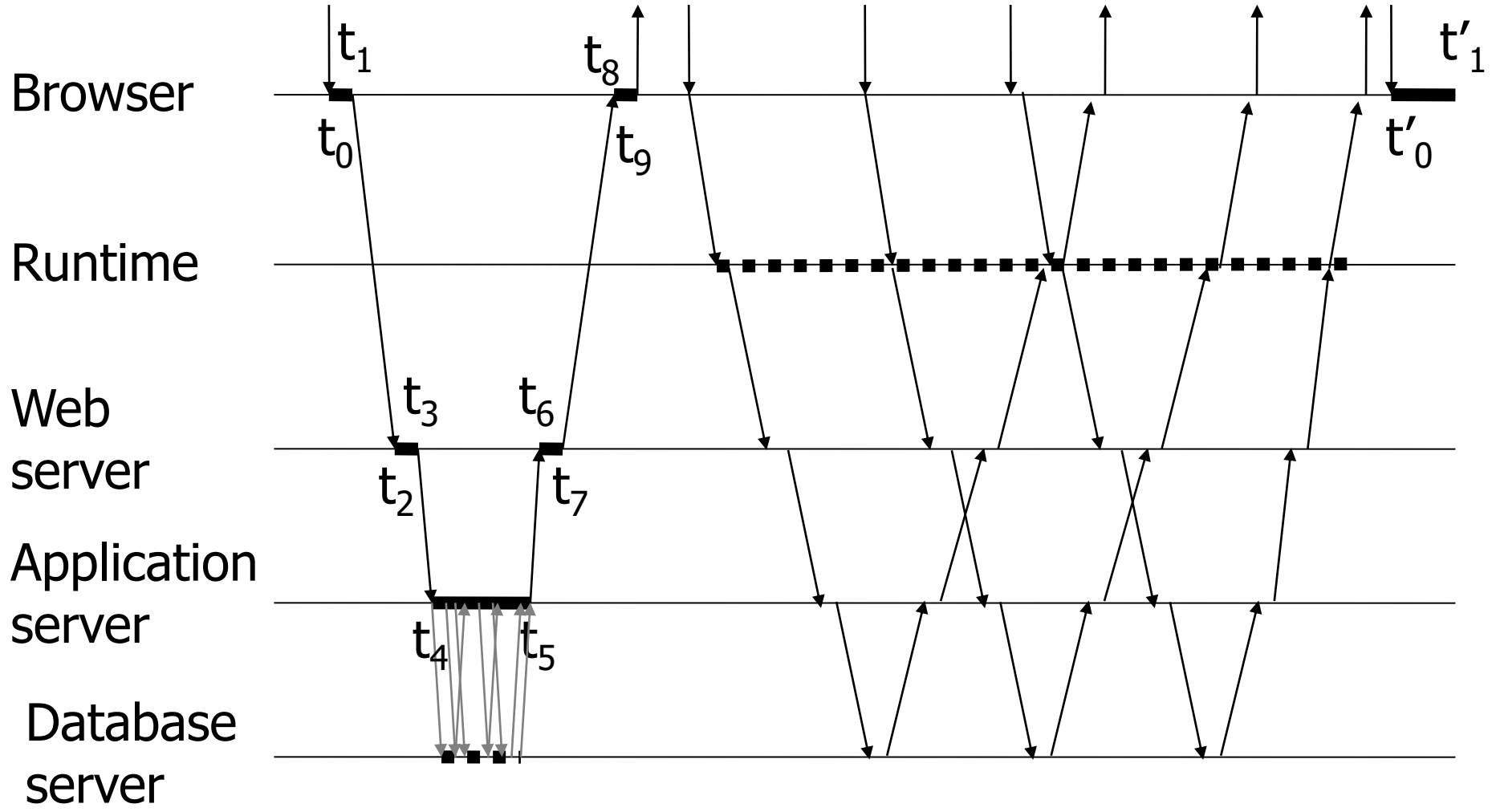
Single Page Applications (SPA)



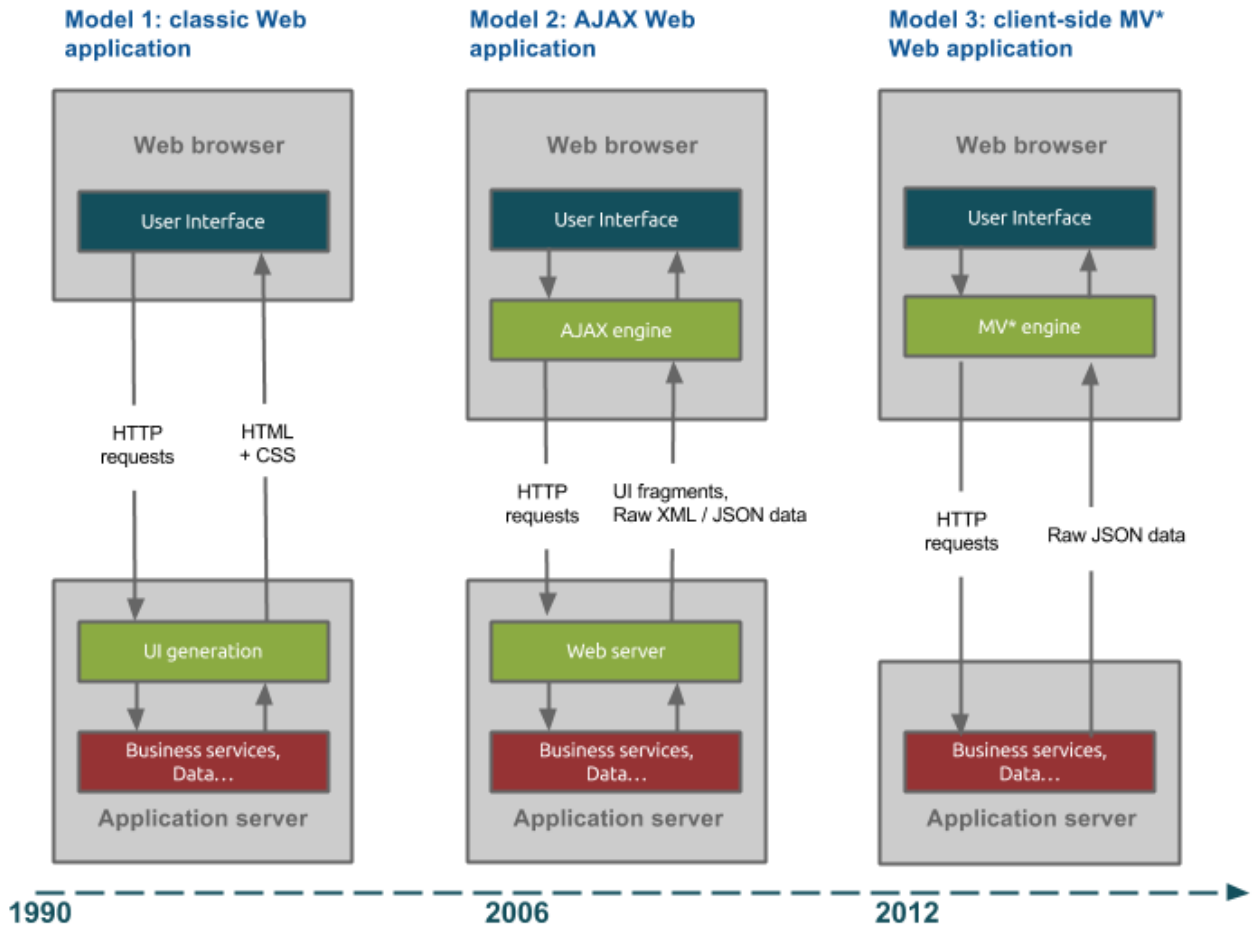
Adopted standards

- AJAX: Asynchronous Javascript and XML
  - XMLHttpRequest for asynchronous communication to the server
  - Data transfer formats: JSON, XML, RDF, RSS, Atom, FOAF, …
- More recently: Fetch API and Promises

Rich-client transaction

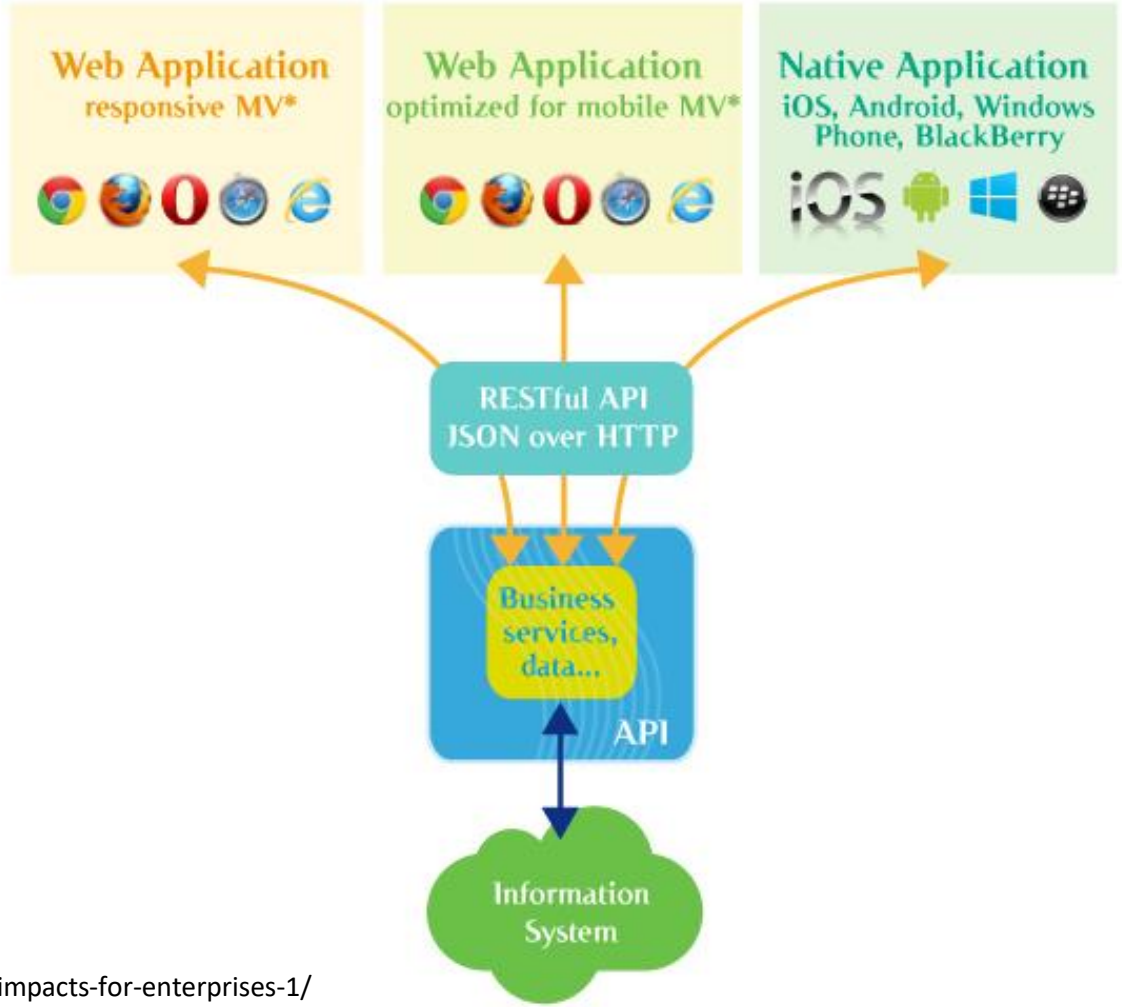


Web application architectures



http://blog.octo.com/en/new-web-application-architectures-and-impacts-for-enterprises-1/

Supporting mobile development



http://blog.octo.com/en/new-web-application-architectures-and-impacts-for-enterprises-1/

Client-side, server-side, databases

Programming languages used in most popular websites*

Websites	Popularity (unique visitors per month)[1]	Front-end (Client-side)	Back-end (Server-side)	Database
Google.com[2]	1,600,000,000	JavaScript	C, C++, Go,[3] Java, Python	BigTable,[4] MariaDB[5]
Facebook.com	1,100,000,000	JavaScript	Hack, PHP (HHVM), Python, C++, Java, Erlang, D,[6] Xhp,[7] Haskell[8]	MariaDB, MySQL,[9] HBase Cassandra[10]
YouTube.com	1,100,000,000	JavaScript	C, C++, Python, Java,[11] Go[12]	BigTable, MariaDB[5][13]
Yahoo	750,000,000	JavaScript	PHP	MySQL, PostgreSQL[14]
Amazon.com	500,000,000	JavaScript	Java, C++, Perl[16]	Oracle Database[17]
Wikipedia.org	475,000,000	JavaScript	PHP, Hack	MySQL[*citation needed*], MariaDB[18]
Twitter.com	290,000,000	JavaScript	C++, Java, Scala, Ruby on Rails[19]	MySQL[20]
Bing	285,000,000	JavaScript	ASP.NET	Microsoft SQL Server
eBay.com	285,000,000	JavaScript	Java,[21] JavaScript[22]	Oracle Database
MSN.com	280,000,000	JavaScript	ASP.NET	Microsoft SQL Server
Microsoft	270,000,000	JavaScript	ASP.NET	Microsoft SQL Server
Linkedin.com	260,000,000	JavaScript	Java, JavaScript,[23] Scala	Voldemort[24]
Pinterest	250,000,000	JavaScript	Django (a Python framework),[25] Erlang	MySQL, Redis [26]
WordPress.com	240,000,000	JavaScript	PHP, JavaScript [27] (Node.js)	MariaDB, MySql

https://en.wikipedia.org/wiki/Programming_languages_used_in_most_popular_websites

References

- HTTP/1.x vs. HTTP/2 – The Difference Between the Two Protocols Explained - https://cheapsslsecurity.com/p/http2-vs-http1/
- How Browsers Work: Behind the scenes of modern web browsers - https://www.html5rocks.com/en/tutorials/internals/howbrowserswork/
- Inside look at modern web browser
  - Part 1: https://developers.google.com/web/updates/2018/09/inside-browser-part1
  - Part 2: https://developers.google.com/web/updates/2018/09/inside-browser-part2
  - Part 3: https://developers.google.com/web/updates/2018/09/inside-browser-part3
  - Part 4: https://developers.google.com/web/updates/2018/09/inside-browser-part4

License

- These slides are distributed under a Creative Commons license “**Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)**”
- **You are free to:**
  - **Share** — copy and redistribute the material in any medium or format
  - **Adapt** — remix, transform, and build upon the material
  - The licensor cannot revoke these freedoms as long as you follow the license terms.
- **Under the following terms:**
  - **Attribution** — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
  - **NonCommercial** — You may not use the material for commercial purposes.
  - **ShareAlike** — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.
  - **No additional restrictions** — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.
- https://creativecommons.org/licenses/by-nc-sa/4.0/