World Wide Web

Written by Febogamedev

The English word web means “spiderweb” and recalls the idea of a very large set of resources connected to one another through hypertext links, called links.
Each link points to a web address, normally expressed through a ***URL***, which identifies a resource and indicates how to reach it through the Internet (Berners-Lee et al., 2005).

Resources may include text, images, videos, documents, data, services, and interactive applications.

- The Web uses the Internet as its communication infrastructure: the Internet is the global network that allows devices and applications to exchange data. The Web is one of the main systems built on top of the Internet and makes it possible to identify, connect, request, and use resources (Berners-Lee, 1990; Fielding et al., 2022).
- The Web is a distributed system: its resources are not all stored on a single computer, but are managed by numerous servers located in different places (Berners-Lee, 1990).

A web page is a document or interface that the browser builds by processing one or more resources. It may contain links to other pages, to different sections of the same page, or to other types of resources.

The three fundamental technologies used to create web pages are:

- ***HTML***, which describes the content, structure, and meaning of the elements. A web page is generally based on an HTML document, which is the main document processed by the browser.
- ***CSS***, which controls the appearance, layout, and adaptation of the page to different devices through rules that define the page style.
- **JavaScript**, which can add behaviors, processing, and interactive functions.

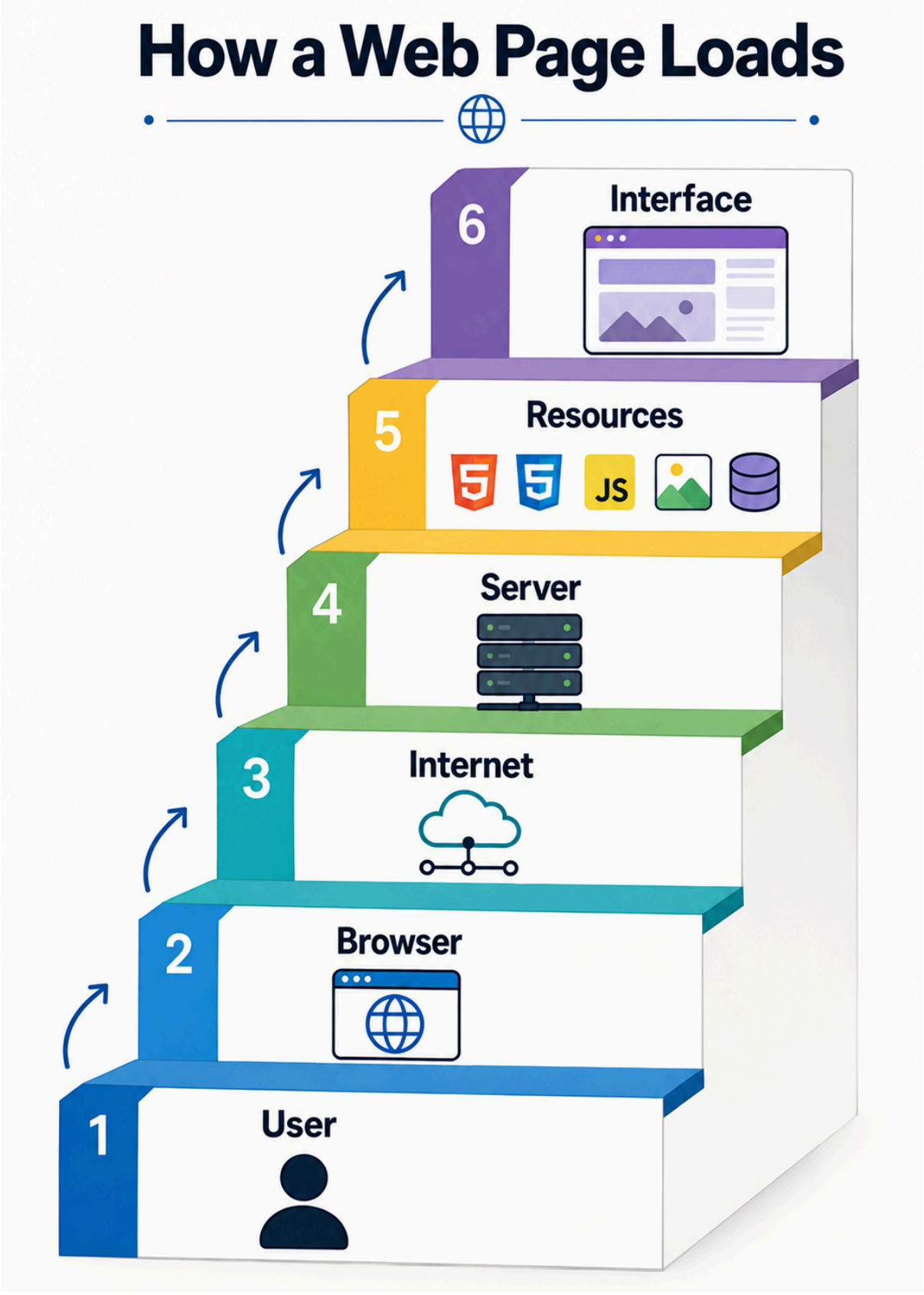
HTML provides the basic structure of the page. CSS and JavaScript are not essential for the existence of a web page, but they make it possible, respectively, to control its presentation and to add more complex behaviors (CSS Working Group, 2026; Ecma International, 2026; WHATWG, 2026).

A website, on the other hand, is an organized collection of related pages and resources, generally associated with the same project, service, or organization and reachable through one or more web addresses.

To access web resources, the user normally uses a client software application called a **browser**. The browser interprets the requested address, communicates with servers, receives data, and processes it to build the interface with which the user interacts (WHATWG, 2026).

Communication between the browser and the web server mainly takes place through ***HTTP***, an application protocol based on the exchange of requests and responses.

Today, this communication is normally protected through ***HTTPS*** (Fielding et al., 2022).



Hypertext

A hypertext is an information structure in which a document contains links to other resources or to different parts of the same document (Berners-Lee, 1990).

Unlike a text designed to be read in a linear order, a hypertext allows reading that is not necessarily sequential.
The reader can choose which links to follow and thus build their own path through the information.

The mechanism that connects an element in the document to a destination is called a **hypertext link**, or simply a **link** (Berners-Lee, 1990).

A hypertext link connects two elements:

- An origin, meaning the selectable element from which the link starts.
- A destination, meaning the resource or section indicated by the link.

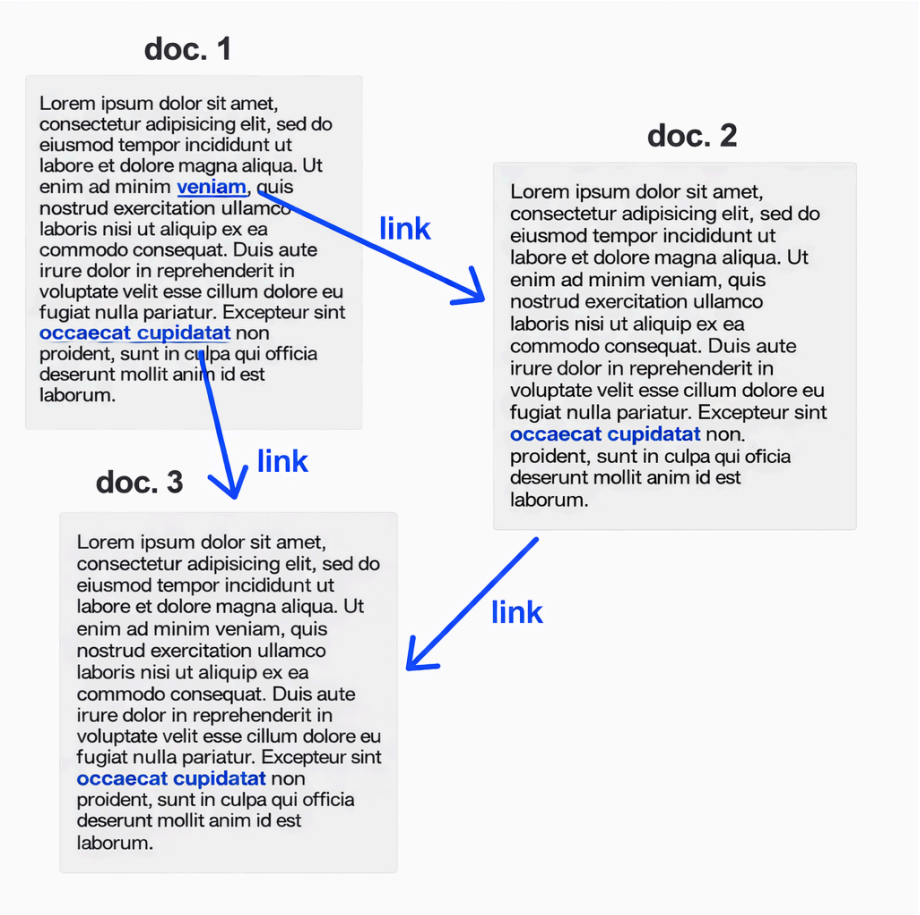
The destination may be:

- Another page on the same website.
- A page belonging to another website.
- A section of the same page or another page.
- An image, a video, a document, or another type of resource.

The origin of the link does not necessarily have to be a single word.
A link can be associated with a sentence, an image, an icon, a menu item, or another interface component.

When links connect not only texts, but also images, videos, audio, and interactive content, we speak of **hypermedia** (Berners-Lee, 1990).
The Web can therefore be considered not only a hypertext system, but also a **hypermedia system**.

The Web was conceived as a large system of connected resources.
A link may lead to a resource hosted on the same server or on a server located elsewhere on the network (Berners-Lee, 1990).
Web navigation is the process through which the user moves among pages, sections, and resources using links, menus, and other orientation systems.



Esempio di ipertesto

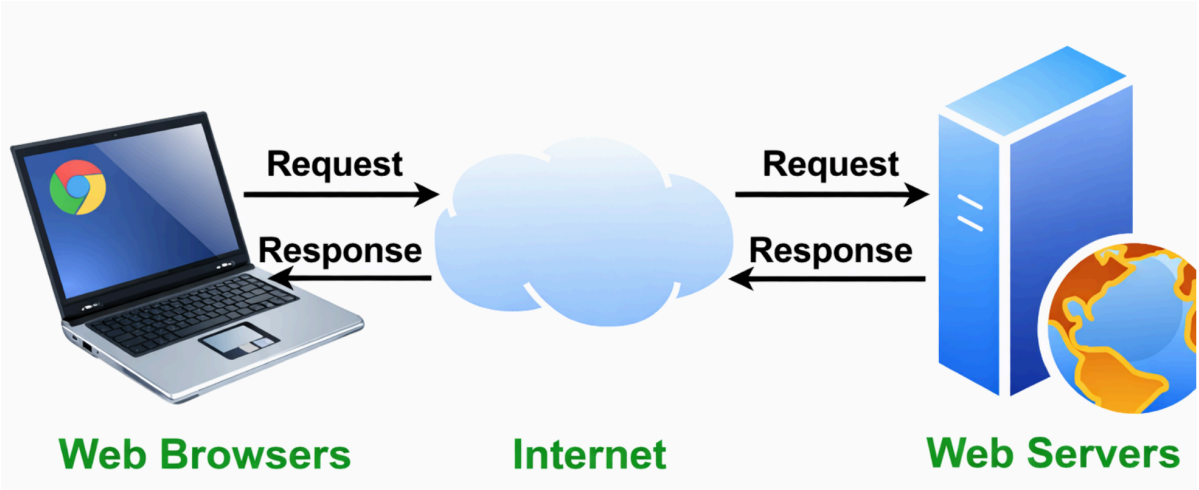
Web server

A **web server** is software that receives *HTTP* requests from clients, such as browsers, and returns the corresponding responses (Fielding et al., 2022).

In common usage, the term *web server* may also refer to the computer or infrastructure on which this software runs. In this chapter, however, we will mainly use the term to refer to the software that manages web communication.

When a browser requests a page, an image, or another type of content, it sends a request to the web server. The server interprets the request, locates or generates the data associated with the requested resource, and returns a response that the browser or another application can interpret (Fielding et al., 2022).

From the server's point of view, the returned content may be **static** or **dynamic**.



Dal punto di vista del server, il contenuto restituito può essere statico o dinamico.

Static resource
A static resource is sent by the server in the form in which it has already been prepared and stored.
Common static resources include:

- Images.
- Audio and video files.
- Fonts.
- CSS style sheets.
- JavaScript files.

The term static means that the server does not need to generate the content specifically for each request, but can send it in the form in which it has been stored.

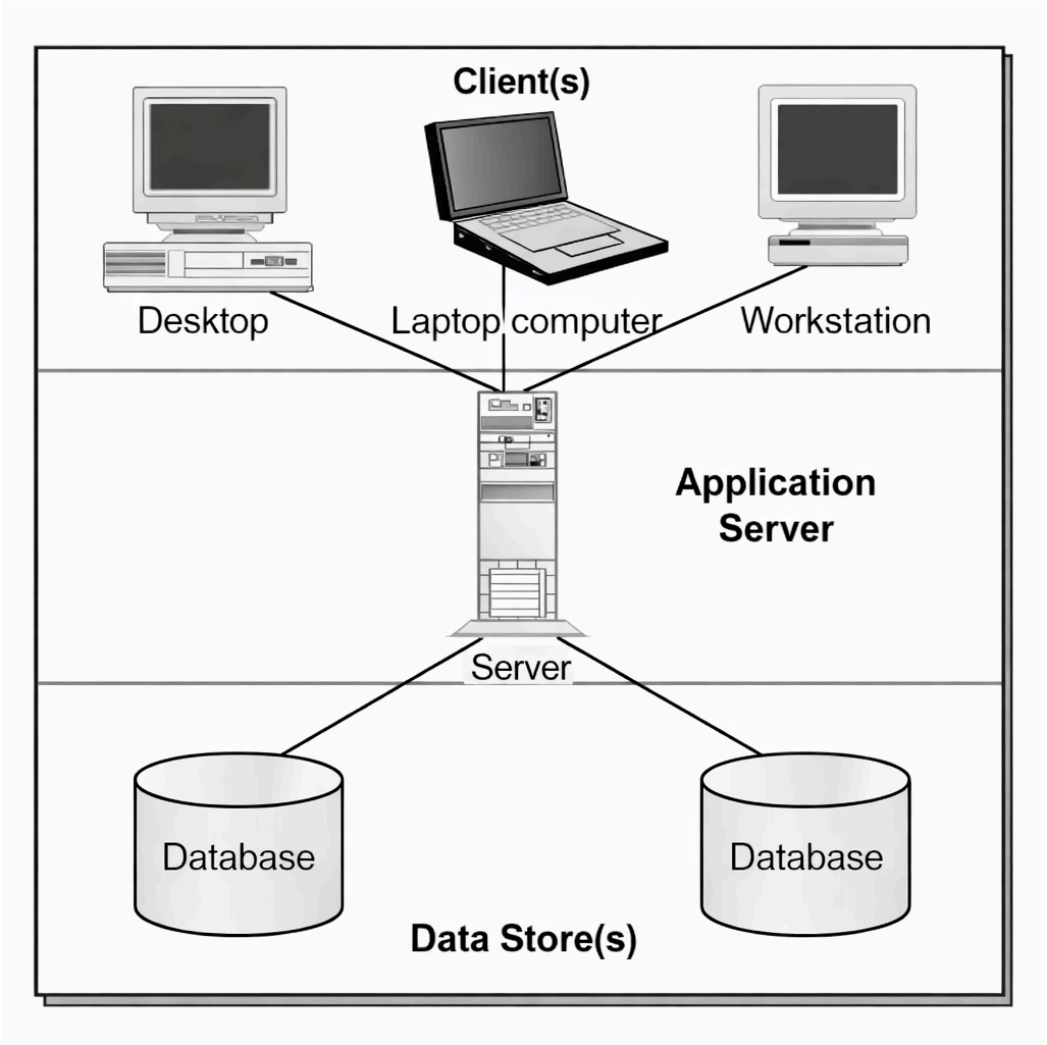
A static resource can still contribute to building an interactive page.
For example, a JavaScript file may be static from the server's point of view, even though it produces dynamic behaviors when executed in the browser (Ecma International, 2026; WHATWG, 2026).

Dynamic resource
A dynamic resource is content generated or assembled by the server in response to a request.

The returned content may depend, for example, on the requested page, the authenticated user, the data entered in a form, the selected language, the products available, information stored in a database, the time, or other conditions.

To generate the response, the web server can work with a **server-side application**, which executes the necessary logic.

The application can communicate with a database or other services, retrieve the required data, and insert it into an HTML document before sending it to the browser.



Example
Consider an e-commerce website:

- When the browser requests the file logo.png, the server retrieves the image already stored in its storage space and sends it without modifying it. The logo is therefore a static resource.
- When, instead, the browser requests the catalog page, a server-side application may query a database, retrieve the available products, prices, and images, and assemble a specific response. In this case, the catalog page is a dynamic resource, because it is generated on the server in response to the request.

Web server software must run on a physical computer or in a virtual environment: a system created through software that behaves like an independent computer.
One of the environments that can host it is a **VPS**, short for Virtual Private Server.
A VPS is a virtual machine created inside a physical server through a virtualization system. The same physical computer can host multiple VPSs, but each one functions as a separate environment, with its own operating system and assigned resources such as CPU, memory, and storage space (Scarfone et al., 2011).

A VPS can host web server software together with the application, database, and other programs required for the website to function.
The VPS therefore represents the virtual environment in which the programs run, while the web server is the software that receives HTTP requests and returns responses (Fielding et al., 2022; Scarfone et al., 2011).

Example
A company may purchase a VPS, install the Linux operating system on it, and configure Nginx as the web server.

When a user visits the website, the request reaches the VPS and is handled by Nginx, which either returns a static resource directly or forwards the request to a server-side application.

Web Application

A web application, or **web app**, is software that the user accesses through a browser.

Unlike native or desktop applications, which normally have to be downloaded and installed on the device, a web app can be used by reaching a web address, although some web apps can also be installed on the device while continuing to use browser technologies and the browser execution environment (WHATWG, 2026; W3C Web Applications Working Group, 2026).

When the user opens a web app, the browser downloads the required resources from the server, such as HTML documents, CSS style sheets, images, JavaScript code, and other data.
The browser interprets these resources and runs the application within its own environment (WHATWG, 2026).

The processing required for the application to function may take place in the browser, on the server, or be distributed between the two.
We can therefore distinguish among:
- Predominantly client-side applications.
- Predominantly server-side applications.
- Hybrid applications.

Predominantly client-side web application
In a predominantly client-side web application, much of the logic and processing is executed by the browser on the user's device (Ecma International, 2026; WHATWG, 2026).

The server initially provides the application files, such as JavaScript code, images, and other resources.

After these resources have been loaded, many operations can be performed locally using the device's CPU and, when possible, GPU.

The server may still be involved in operations such as authentication, synchronization, online saving, collaboration, or access to external services.

Main advantages:

- Fast interaction, because many operations do not require continuous data exchange with the server.
- Lower computational load on the server.
- The possibility of processing data and files directly on the device.
- Some functions can be used without a connection, if the application has been designed to work offline and the necessary resources have been stored locally (W3C Web Applications Working Group, 2026).
-

Main disadvantages:

- Performance depends on the power of the device.
- The browser limits direct access to the operating system and files.
- Initial loading may require transferring a large amount of code and resources.
- Code executed in the browser is sent to the user's device and cannot contain secret information or be considered trustworthy for authorizing important operations (OWASP Foundation, 2021).

Example
Piskel is a web app for creating pixel art, sprites, and animations that runs mainly in the browser.

When the user opens Piskel, the browser downloads the HTML, CSS, and JavaScript files required for the application to function.
Operations such as drawing pixels, managing frames, displaying the animation preview, and generating GIF files or spritesheets are processed mainly on the user's device.
The server initially provides the application files, but it does not need to be contacted every time the user draws a pixel or modifies a frame.

Predominantly server-side web application
In a predominantly server-side web application, a significant part of the logic, data management, and processing is executed by the server.
The browser continues to manage the display of the interface and collect the user's actions, while the server can:

- Verify the user's identity and permissions.
- Retrieve information from a database.
- Perform calculations or processing.
- Generate an HTML document or other data.
- Record the operations performed.
- Send the result to the browser (Fielding et al., 2022).

Main advantages:

- The most important data and logic remain under the operator's control.
- The application depends less on the power of the user's device.
- Server-side logic can be updated centrally.
- It is easier to share and synchronize data among users and devices.

Main disadvantages:
- Many functions depend on the connection.
- The server must support the users' computational load.
- Response times may increase when the network is slow.
- Infrastructure management can be expensive.

Example
When a user accesses their bank account, the browser displays the interface and allows them to issue commands.
The server verifies credentials and permissions, retrieves the balance, records operations, and authorizes any transfers.
These operations must be verified by the server because the browser runs on a device controlled by the user and cannot be considered a trustworthy source.
The server therefore maintains the official state of the account and decides which operations can be performed.

Hybrid web applications
In practice, many modern web applications are hybrid.
The browser can manage the interface, animations, immediate data validation, and part of the processing.
The server can instead handle authentication, databases, synchronization, restricted operations, and services that require large computational resources.
The distinction between client-side and server-side therefore does not indicate completely separate categories, but describes where the different parts of the application are executed.

Example
Photopea is a web app for image editing that runs mainly in the browser.
When the user opens Photopea, the browser downloads the required code and resources.
Operations such as using brushes, managing layers, selections, filters, and edit history are processed mainly by the user's device.
- Local files can be opened, processed, and saved without being transferred to Photopea's servers.
- The application can still communicate with external services when the user chooses to use cloud storage systems. 1_World Wide Web

URL

A web resource can be identified through an address, generally a ***URL***.
URL stands for Uniform Resource Locator.

A URL is a standardized string that identifies a resource and provides the information required to refer to it (Berners-Lee et al., 2005).

URLs make it possible to reach web pages, images, videos, documents, files, services, and other resources available through the Web.
Different URLs can also lead to the same resource.

To represent in simplified form an absolute URL that uses HTTP or HTTPS, we can use the following structure. Elements in square brackets are optional (Berners-Lee et al., 2005):

schema://host[:port][/path][?query][#frammento]

Example
https://example.com/catalog/shoes?color=black&size=42#models

Its components are:
- Scheme: https.
- Host: example.com.
- Path: /catalog/shoes.
- Query: color=black&size=42.
- Fragment: models.

Scheme
The scheme is the first part of the URL and ends with the : character.

Its purpose is to indicate the type of URL and the rules according to which it must be interpreted.
In URLs with the http and https schemes, the scheme also indicates the communication system to be used (Berners-Lee et al., 2005; Fielding et al., 2022).

The scheme is normally written in lowercase.
An absolute URL must include the scheme.

The user may type only, for example: example.com
In this case, the user is not formally entering a complete URL.
The address bar interprets the text and can automatically construct a URL, generally by trying to use *HTTPS*.
The exact behavior depends, however, on the browser and its settings.

In URLs with the http and https schemes, the : character ends the scheme, while // introduces **the authority**, the part that identifies the entity that manages or hosts the resource and that, in the simplified form, includes the host and the optional port.

Not all schemes use an authority component.
For example:
mailto:name@example.com

In this case, mailto is the scheme and is not followed by //.
The browser may use the address to open the configured email application (IANA, n.d.-b).
Some common schemes are (IANA, n.d.-b):

Nome	Utilizzo
http	Access to resources through HTTP.
https	Access to resources through HTTP over a connection protected by TLS.
ftp	File transfer through FTP; today it is generally not handled directly by browsers.
about	Access to internal browser pages or information.
file	Indicates that the resource is located in the device's local file system. The browser can therefore be used to open and display certain local files, such as documents, images, or HTML pages.
mailto	Opens the email application to write to an address.

Host
The host identifies the network destination to which the request is directed.
It may be represented by a domain name or an IP address.

Example with a domain name: https://example.com
Example with an IPv4 address: http://192.0.2.10

When a domain name is used, DNS allows the browser to obtain the IP address associated with the specified host (Mockapetris, 1987).

Port
The port identifies the access point to the network service with which the client must communicate.
The http and https schemes have default ports:

- The protocol associated with HTTP normally uses port 80.
- The protocol associated with HTTPS normally uses port 443 (IANA, n.d.-a).

When the default port is used, the number may be omitted.
The following two URLs therefore refer to the same port:

- https://example.com.
- https://example.com:443.

A different port must instead be specified explicitly, for example:
https://example.com:8443

Path
The path identifies a target within the addressing space managed by the server or application (Berners-Lee et al., 2005).

For example:
/catalog/shoes
The path may resemble a folder and subfolder structure, but it does not necessarily correspond to folders or files that actually exist in the server's file system.

On a static website, a path may correspond directly to a file.
For example: /images/logo.svg

In a dynamic application, on the other hand, a path such as "/catalog/shoes" may be interpreted by server-side software as an instruction to retrieve some data and generate the response.

The path therefore identifies the target of the request, but its concrete meaning is established by the server or application (Nottingham, 2020).
In URLs with the http and https schemes, when a path is not explicitly written, the root path / is used after the authority declaration.

In this case, the server decides which content to associate with that path.
On a static website, it may be configured to return a document such as index.html, but this is a server configuration and not a universal rule of URLs.

Example
https://example.com/
In this URL, no path is explicitly specified; the server will therefore contain a rule that associates the root path with a particular web resource.

Example
https://example.com/prodotti/scarpe
In this URL:
- The server's file system might contain a products folder and, inside it, a shoes folder containing an index.html file.
- Alternatively, there may be a rule instructing the system to search the database for all products belonging to the "shoes" category and build a page with the results.

Query string (opzionale)
The query, often called the query string, is the part of the URL introduced by the ? symbol.

It makes it possible to add information or parameters to the request (Berners-Lee et al., 2005).

Example: https://example.com/catalog?color=black&size=42

A very common convention is to use pairs consisting of a parameter name and its corresponding value:
?parameter1=value1&parameter2=value2

The = character separates the name from the value, while & separates the different pairs. This is a very common convention, but the meaning and structure of the query are determined by the application (Nottingham, 2020).

In the previous example, the parameters might indicate that the catalog should be filtered to display black products in size 42.

The query can be used to:

- Filter a list.
- Perform a search.
- Choose a language.
- Sort results.
- Indicate a page.
- Store or share part of the interface state.

It is important to specify that when an HTTP or HTTPS request is made to the server, the query is normally included in the request that is sent (Berners-Lee et al., 2005; Fielding et al., 2022).
That request can be processed by the server or, in some applications, by a JavaScript program running in the browser.

Example
The following URL belongs to an online public notice board service:
https://web.spaggiari.eu/sdg/app/default/albo_pretorio.php?sede_codice=MIIT0027

The part “sede_codice=MIIT0027” is the query string.
It contains a parameter called sede_codice, to which the value MIIT0027 is assigned.

The application receives this parameter and may use it to determine which public notice board to display.
For example, it could use it to retrieve the data associated with the specified institution and generate the response to send to the browser.

Fragment
The fragment is the part of the URL introduced by the # character.
Example: https://example.com/manual#installation

The fragment makes it possible to identify a specific part or view of the resource indicated by the rest of the URL (Berners-Lee et al., 2005).

Unlike the query, the fragment is not sent to the server.
It is separated from the rest of the URL and interpreted by the browser or by the client-side application (Berners-Lee et al., 2005).
In an HTML document, the fragment may correspond to an element with a specific id attribute.

Suppose our HTML document contains this piece of code: <h2 id="installation">Installation</h2>

When we request this URL from the server: https://example.com/manual#installation

the browser downloads the /manual resource, looks for the element with: id="installation" and generally brings it into the visible area.
This behavior is particularly useful on very long pages, where the user does not want to scroll all the way to the desired section (WHATWG, 2026).

In common usage, a link containing a fragment is often called an anchor **link** (Berners-Lee et al., 2005).
More precisely, however:

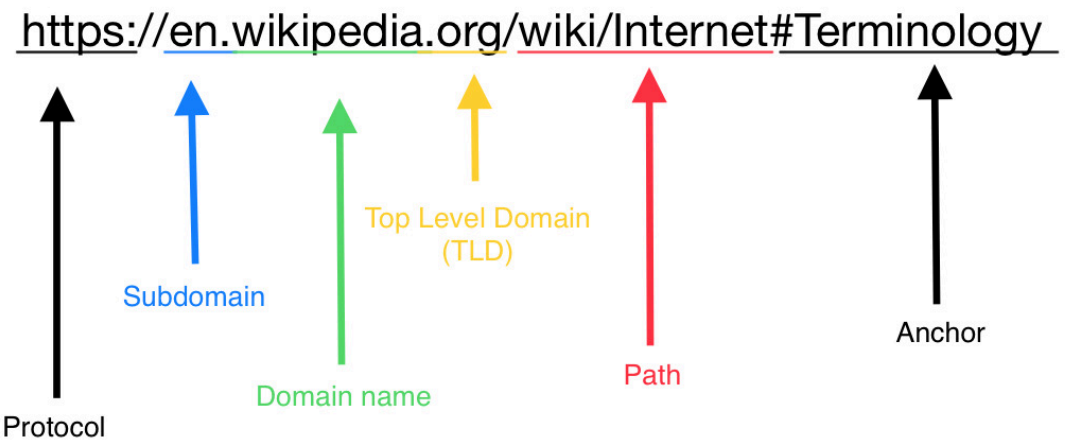
- The fragment is the URL component.
- The destination anchor is the point or element identified within the document.

Example
he Wikipedia page about the Internet is very long and divided into several sections.
A URL containing a fragment can allow the browser to jump directly to a specific section.

For example:
https://en.wikipedia.org/wiki/Internet#Terminology

The part following the # symbol is the URL fragment.

On an HTML page, the browser normally looks for an element whose id attribute corresponds to the text following # and, if it finds one, brings that part of the document into the visible area.


https://en.wikipedia.org/wiki/Internet#Terminology
Protocol
Subdomain
Domain name
Top Level Domain
(TLD)
Path
Anchor

Supplementary material

Exercises

Are the Internet and the Web the same thing?
One student claims:
“The Internet and the World Wide Web are two different names for the same thing.”

Another replies:
“No. The Internet is the network, while the Web is one of the systems that uses that network.”

Analyze the two statements.

- Explain which of the two descriptions is more accurate according to the handout.
- Describe the function performed by the Internet.
- Describe the function performed by the World Wide Web.
- Explain why the Web can be considered a distributed system.
- Name at least three types of resources that can be available through the Web.
- Imagine that the Internet continues to work but all web servers temporarily become unreachable. Explain why this would not necessarily mean that the entire Internet had stopped working.
- Finally, explain why the browser must use the Internet to reach a web resource, while the Internet is neither the browser nor the Web.

Designing hypertext navigation
A school wants to publish a guide for new students on its website.
The guide contains:
Home → About the school → Study programs → Laboratories → Regulations → Contacts

However, the school wants to avoid forcing students to read the pages in this order.
It also wants to allow them to:

- Go directly from the page of a study program to the related laboratories.
- Open a PDF document containing the regulations.
- Move from the contacts page to the transport company’s website.
- Jump directly to the “Timetables” section on the same contacts page.

Design the structure conceptually:

- Explain why the guide can be organized as a hypertext.
- Identify at least four hypertext links you would include.
- For each one, identify its origin and destination.
- Distinguish a link to another page on the same website, one to an external website, and one to a section of the same page.
- Explain why the origin of a link does not necessarily have to be a word.
- If the guide also links videos, images, and audio, explain why we can speak of hypermedia.

- Explain why a hypertext structure allows the user to build different navigation paths instead of necessarily following a predefined sequence.

What happens when I open an e-commerce website?
A user opens an online store and visits the page:
https://shop.example.com/catalog

The page displays the company logo, the website style sheet, and a product catalog that changes according to the availability stored in the database.

Reconstruct what happens.
- Explain the role of the browser.
- Explain the role of the web server.
- Identify the protocol mainly used in the exchange of requests and responses between browser and server.
- Explain why HTTPS is normally used today.
- Classify the logo, the CSS file, and any JavaScript file already stored on the server as static or dynamic resources.
- Explain why a JavaScript file can be static from the server's point of view but still produce dynamic behaviors in the browser.
- Explain why the catalog page can instead be considered a dynamic resource.
- Reconstruct the possible process: browser request → web server → server-side application → database → response generation → browser.
- Explain why the page displayed to the user can be built using HTML, CSS, JavaScript, images, and dynamically generated data at the same time.

Web server, physical server, or VPS?
A small company wants to publish its website.
It purchases a VPS, installs Linux, and then installs Nginx.
A server-side application and a database also run on the same VPS.
An employee states:
"The VPS is the web server."

Analyze the statement.
- Explain what a VPS is.
- Explain what a web server is according to the main meaning used in the handout.
- Identify the role played by Nginx in the example.
- Explain why VPS and web server are not synonyms.
- Explain how multiple VPSs can be hosted on the same physical computer.
- Indicate which resources can be assigned separately to a VPS.
- Reconstruct the path of a request: user → Internet → VPS → Nginx → possible application → response.
- Explain why the same VPS can simultaneously host a web server, an application, and a database without these terms meaning the same thing.

Where should the program run?
You need to design three web applications.

- Application A: a simple pixel-art editor. Drawing must react immediately to mouse movement and can be saved on the device.
- Application B: a banking service. It must control the balance, user identity, permissions, and money transfers.
- Application C: a collaborative editor. Some operations are executed locally, but documents and accounts must be synchronized across multiple devices..

For each application:
- Decide whether a predominantly client-side, predominantly server-side, or hybrid architecture would be more suitable.
- Justify your choice.
- Indicate which operations you would have the browser perform.
- Indicate which operations you would instead assign to the server.
- Explain why, in application A, it may be advantageous to perform many operations directly on the device.
- Explain why, in application B, the browser should not be able to decide autonomously whether a bank transfer is authorized.
- Explain why JavaScript code sent to the browser should not contain secret information.
- Explain why application C is a good example of a hybrid application.
- Conclude by explaining why “client-side” and “server-side” mainly describe where processing takes place, rather than two completely separate categories of websites.

Breaking down a URL
Consider this URL:
https://shop.example.com:8443/catalog/shoes?color=black&size=42#models

Analyze it completely.
- Identify the scheme, host, port, path, query, and fragment.
- Explain the function of the https scheme.
- Explain the function of the host shop.example.com.
- Describe what must happen if the browser knows the host name but not its IP address.
- Explain why port 8443 must be written explicitly, while port 443 can normally be omitted from an HTTPS URL.
- Explain why /catalog/shoes does not necessarily correspond to two folders that actually exist on the server’s disk.
- Explain how a dynamic application might interpret that path.
- Interpret the parameters color=black and size=42.
- Explain the difference between the ? symbol and the # symbol.
- Indicate which part, query or fragment, is normally sent to the server.
- Explain what the browser might do with the fragment #models.

Now imagine changing only the fragment:
https://shop.example.com:8443/catalog/shoes?color=black&size=42#prices

Explain why the browser might not need to request a different resource from the server.

Bibliography

Berners-Lee, T. (1990). *Information management: A proposal*. CERN.

Berners-Lee, T., Fielding, R., & Masinter, L. (2005). *Uniform Resource Identifier (URI): Generic syntax* (RFC 3986). Internet Engineering Task Force.

CSS Working Group. (2026). *CSS snapshot 2026*. World Wide Web Consortium.

Ecma International. (2026). *ECMAScript 2026 language specification* (ECMA-262, 17th ed.).

Fielding, R., Nottingham, M., & Reschke, J. (Eds.). (2022). *HTTP semantics* (RFC 9110). Internet Engineering Task Force.

Internet Assigned Numbers Authority. (n.d.-a). *Service name and transport protocol port number registry*. Retrieved August 27, 2026.

Internet Assigned Numbers Authority. (n.d.-b). *Uniform Resource Identifier (URI) schemes*. Retrieved August 27, 2026.

Mockapetris, P. (1987). *Domain names—concepts and facilities* (RFC 1034). Internet Engineering Task Force.

Nottingham, M. (2020). *URI design and ownership* (RFC 8820). Internet Engineering Task Force.

OWASP Foundation. (2021). *A01:2021—Broken access control*.

Scarfone, K., Souppaya, M., & Hoffman, P. (2011). *Guide to security for full virtualization technologies* (NIST SP 800-125). National Institute of Standards and Technology.

WHATWG. (2026). *HTML Living Standard*.

W3C Web Applications Working Group. (2026). *Service workers*. World Wide Web Consortium.

Further reading

- James F. Kurose and Keith W. Ross, Computer Networking: A Top-Down Approach, 9th ed., Pearson, 2026 — The best general reference for understanding the Web as an application-layer system built on top of the Internet. It covers client–server architectures, HTTP, DNS, sockets, web applications, content distribution, and the network mechanisms that make browser–server communication possible. The top-down approach is particularly suitable for this handout because it begins with applications before moving toward the lower network layers. Pearson
- Andrew S. Tanenbaum, Nick Feamster, and David J. Wetherall, Computer Networks, 6th ed., Pearson, 2022 — Provides a broader treatment of Internet architecture, distributed applications, the application layer, HTTP, DNS, clients, and servers. Compared with Kurose and Ross, it is especially useful for placing the Web within the overall architecture of computer networks and understanding how the different layers cooperate. Pearson
- Jon Duckett, HTML & CSS: Design and Build Websites, Wiley, 2011 — A highly visual and accessible introduction to HTML documents, hyperlinks, images, page structure, CSS, layout, and the construction of web pages. It is particularly relevant to the section of the handout that explains how HTML provides structure while CSS controls presentation. Wiley Store
- Jon Duckett, JavaScript and jQuery: Interactive Front-End Web Development, Wiley, 2014 — Focuses on the client-side execution model, showing how JavaScript can

react to user actions, modify pages, process information in the browser, and create interactive interfaces. It is therefore a useful extension of the handout's distinction between resources supplied by the server and processing performed locally by the browser. Wiley Newsroom

- Jon Duckett, PHP & MySQL: Server-side Web Development, Wiley, 2022 — Complements the previous titles by focusing on the server side: receiving requests, executing application logic, working with databases, authenticating users, and dynamically generating web content. It closely matches the handout's sections on dynamic resources and predominantly server-side applications.

Editorial notes

In producing these materials, I have used generative artificial intelligence tools to support the writing process, particularly to improve the form of the text, reorganize content, refine wording, and speed up certain editorial tasks.

As I work independently on the production of these materials, I try to automate any activities that can reasonably be automated, so that I can devote more time to research, design, and content development.

However, artificial intelligence does not determine the content of this work: the choice of topics, structure, ideas, interpretations, examples, and teaching approach are developed by me. AI is therefore used primarily as a tool to support production and formal revision, while authorship and responsibility for the design and content remain mine.

Unless otherwise indicated, this material is distributed under the Creative Commons Attribution–NonCommercial–ShareAlike 4.0 International license (CC BY-NC-SA 4.0).

You are therefore free to share, redistribute, adapt, and create derivative works based on this material, provided that appropriate credit is given, the material is not used for commercial purposes, and any modified or derivative versions are distributed under the same license.

