Web Development

CONTENT

- **Web programming fundamentals**
- Working of web browser,
- HTTP protocol,
- HTTPS,
- DNS,
- TLS,
- XML introduction,
- Json introduction, DOM, URL, URI, REST API

What is a Browser?

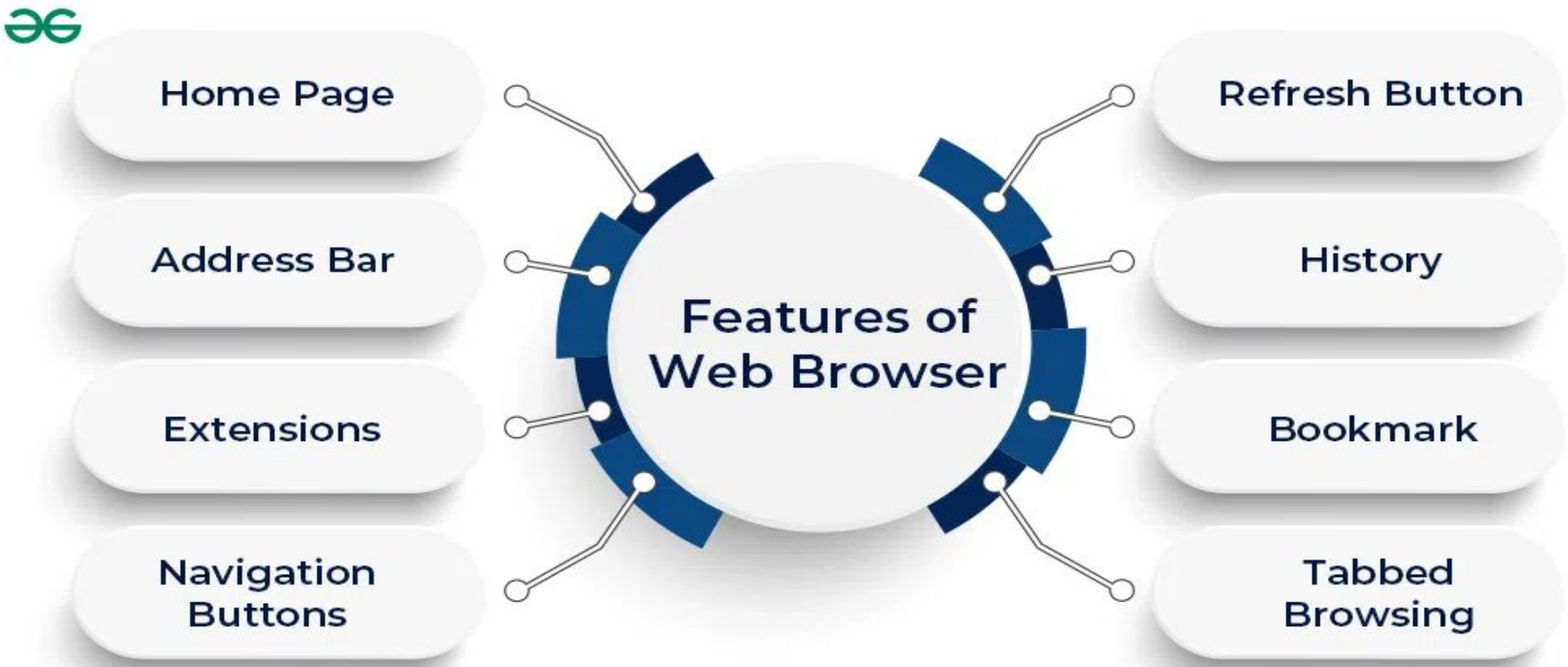
- A browser is a software program that is used to explore, retrieve, and display the information available on the World Wide Web. This information may be in the form of pictures, web pages, videos, and other files that all are connected via hyperlinks and categorized with the help of URLs (Uniform Resource Identifiers).
- A browser is a client program as it runs on a user computer or mobile device and contacts the webserver for the information requested by the user. The web server sends the data back to the browser that displays the results on internet supported devices. On behalf of the users, the browser sends requests to web servers all over the internet by using HTTP (Hypertext Transfer Protocol). A browser requires a smartphone, computer, or tablet and internet to work.

History of Web Browser

1. The **WorldWideWeb** was the first web browser. It was created by W3C Director Tim Berners-Lee in **1990**. Later, it was renamed **Nexus** to avoid confusion caused by the actual World Wide Web.
2. The **Lynx** browser was a text-based browser, which was invented in **1992**. It was not able to display the graphical content.
3. Although, the first graphical user interface browser was NCSA Mosaic. It was the first most popular browser in the world, which was introduced in **1993**.
4. In **1994**, there were some improvements occurred in Mosaic and came to Netscape Navigator.
5. In **1995,** Microsoft introduced the **Internet Explorer** It was the first web browser developed by Microsoft.

6. A research project started on Opera in **1994**. Later, it was publicly introduced in 1996.
7. **Apple's Safari** browser was introduced in **2003**. It was specifically released for Macintosh computers.
8. In **2004**, Mozilla introduced **Firefox** as Netscape Navigator.
9. In **2007**, a browser **Mobile Safari** was released as Apple mobile web browser.
10. The popular browser **Google Chrome** was launched in **2008**.
11. The fast-growing mobile-based browser **Opera Mini** was released in **2011**.
12. The Microsoft **Edge** browser was launched in **2015**.

Features of Web Browser



- **Home Page:** Whenever we open a browser, the first web page which loads is called the home page. Although the home page of a browser has some default setting, it can be changed according to user's preference.

- **Address (URL) Bar:** The URL of the website that has to be accessed is entered here. Through this bar user can access different websites or search anything on browser.

- **Navigation Buttons:** Using the forward and backward buttons user can revisit previously opened web pages. These buttons provide convenience to retrieve data previously accessed.

- **History:** If you visited a website and want to revisit it after a few days but have forgotten the address of website then don't worry as every browser has a history where it collects all previously visited data. This is a very handy feature to revisit websites.

- **Bookmark:** This feature helps in quickly accessing websites. If you come across some important web page you can always bookmark it for easy access in future.
- **Refresh Button:** The refresh button reloads the page and shows updates if any.
- **Home Button:** It brings the user to home page which can be set by user.
- **Tabbed Browsing:** Using this feature a user can open multiple web pages in the same web browser so that they can easily switch between them.
- **Extensions:** Extensions or plugins (like AdBlock) can be installed by the user. These extensions help in improving functionality of web browsers.

Components of a Browser

- A browser has two elements: **front-end** and **back-end**.

- The **frontend** is what users see and interact with,
- while the **backend** is how everything works behind the scenes.

1. Front End Development

- The frontend is the part of the website users directly interact with. This includes the design, menus, text, images, videos, and overall layout. The main languages used for frontend development are HTML, CSS, and JavaScript. Responsiveness and performance are the two main objectives of the Front End. The developer must ensure that the site is responsive i.e. it appears correctly on devices of all sizes no part of the website should behave abnormally irrespective of the size of the screen.

Front End Languages

- **HTML**: HTML stands for Hypertext Markup Language. It is used to design the front-end portion of web pages using a markup language.
- **CSS**: Cascading Style Sheets fondly referred to as CSS is a simply designed language intended to simplify the process of making web pages presentable
- **JavaScript**: JavaScript is a famous scripting language used to create magic on sites to make the site interactive for the user
- There are many other languages through which one can do front-end development depending upon the framework for example **Flutter uses Dart, React uses JavaScript and Django uses Python, and much more**.

2. Back End Development

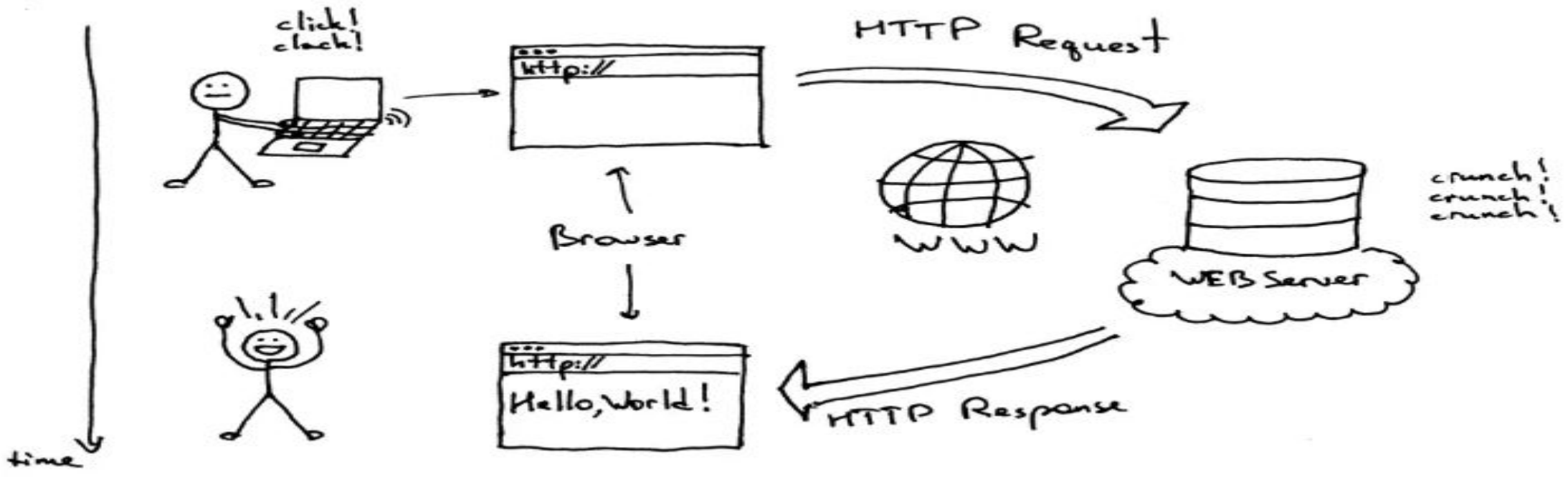
- The backend is the server side of the website. It manages data and ensures everything on the frontend works properly. Users don’t see or interact directly with the backend; it’s the behind-the-scenes functionality. The parts and characteristics developed by backend designers are indirectly accessed by users through a front-end application.

Back End Languages

- **PHP**: PHP is a server-side scripting language designed specifically for web development. Since PHP code is executed on the server side, it is called a server-side scripting language.
- **C++**: It is a general-purpose programming language and is widely used nowadays for competitive programming..
- **Java**: Java is one of the most popular and widely used programming languages and platforms. It is highly scalable
- **Python**: Python is a programming language that lets you work quickly and integrate systems more efficiently
- **Node.js**: Node.js is an open-source and cross-platform runtime environment for executing JavaScript code outside a browser.

How does a Browser Work?

- Browsers are responsible for retrieving and displaying web content to users. When a user enters a URL or clicks on a link, the browser initiates a complex series of actions to retrieve the web content from a server and display it on the user's device.



- The process begins with Domain Name System (DNS) resolution, where the browser translates the domain name into an IP address to locate the server where the web page is stored.
- The browser then sends an HTTP request to the server, specifying the path and parameters of the requested resource.
- Once the server receives the request, it sends an HTTP response to the browser containing the requested resource in HTML, CSS, and JavaScript code.
- The browser’s rendering engine interprets and renders the code to display the web page on the user’s device.
- The CSS stylesheets are applied to format the web page’s content, including fonts, colors, and layout.
- The browser may also execute JavaScript code on the web page to add interactivity and dynamic behavior.
- As new content is loaded or changes are made to the web page, the browser updates the display accordingly.

What is the URL (Uniform Resource Locator)?

- A **uniform resource locator** is the address of a resource on the internet or the World Wide Web. It is also known as a web address or uniform resource identifier (URI).
- For example, **https: www.javatpoint.com**, which is the URL or web address for the javatpoint website. A URL represents the address of a resource, including the protocol used to access it.

Domain Name System (DNS)

- The Domain Name System (DNS) is like the internet's phone book. It helps you find websites by translating easy-to-remember names (like www.example.com) into the numerical IP addresses (like 192.0.2.1) that computers use to locate each other on the internet. Without DNS, you would have to remember long strings of numbers to visit your favorite websites.

- Domain Name System (DNS) is a hostname used for **IP address** translation services. DNS is a distributed database implemented in a hierarchy of name servers. It is an application layer protocol for message exchange between clients and servers. It is required for the functioning of the Internet

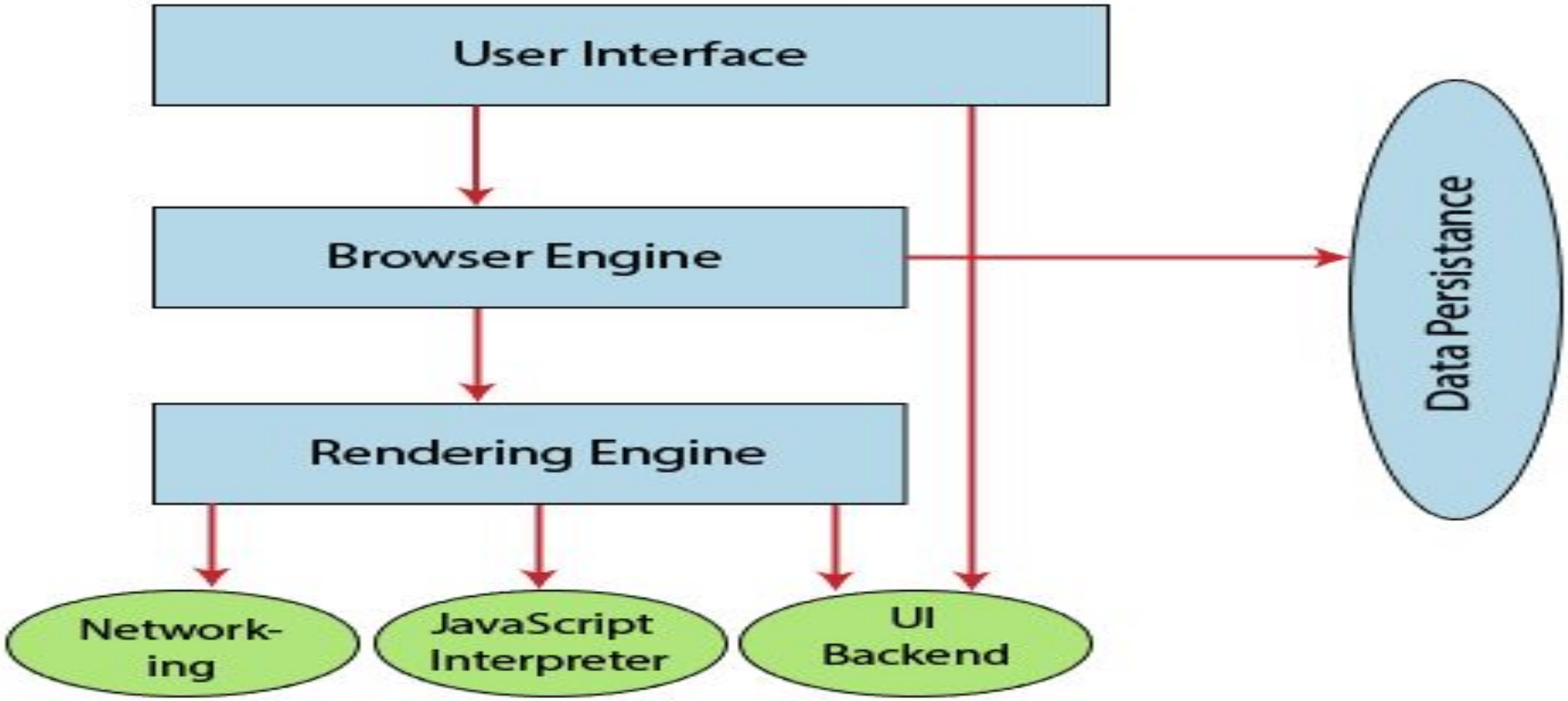
What is the Need for DNS?

- Every host is identified by the IP address but remembering numbers is very difficult for people also the IP addresses are not static therefore a mapping is required to change the domain name to the IP address. So DNS is used to convert the domain name of the websites to their numerical IP address.

Types of Domain

- There are various kinds of domains:
- **Generic Domains:** .com(commercial), .edu(educational), .mil(military), .org(nonprofit organization), .net(similar to commercial) all these are generic domains.
- **Country Domain:** .in (India) .us .uk
- **Inverse Domain:** if we want to know what is the domain name of the website. IP to domain name mapping.

Component of a Web browser



1. **User Interface:** The user interface is an area where the user can use several options like address bar, back and forward button, menu, bookmarking, and many other options to interact with the browser.
2. **Browser Engine:** It connects the UI (User Interface) and the rendering engine as a bridge. It queries and manipulates the rendering engine based on inputs from several user interfaces.
3. **Rendering Engine:** It is responsible for displaying the requested content on the browser screen. It translates the HTML, XML files, and images, which are formatted by using the CSS. It generates the layout of the content and displays it on the browser screen. Although it can also display the other types of content by using different types of plugins or extensions. such as:
    1. Internet Explorer uses **Trident**
    2. Chrome & Opera 15+ use **Blink**
    3. Chrome (iPhone) & Safari use **Webkit**
    4. Firefox & other Mozilla browsers use **Gecko**

4. Networking: It retrieves the URLs by using internet protocols like HTTP or FTP. It is responsible for maintaining all aspects of Internet communication and security. Furthermore, it may be used to cache a retrieved document to reduce network traffic.

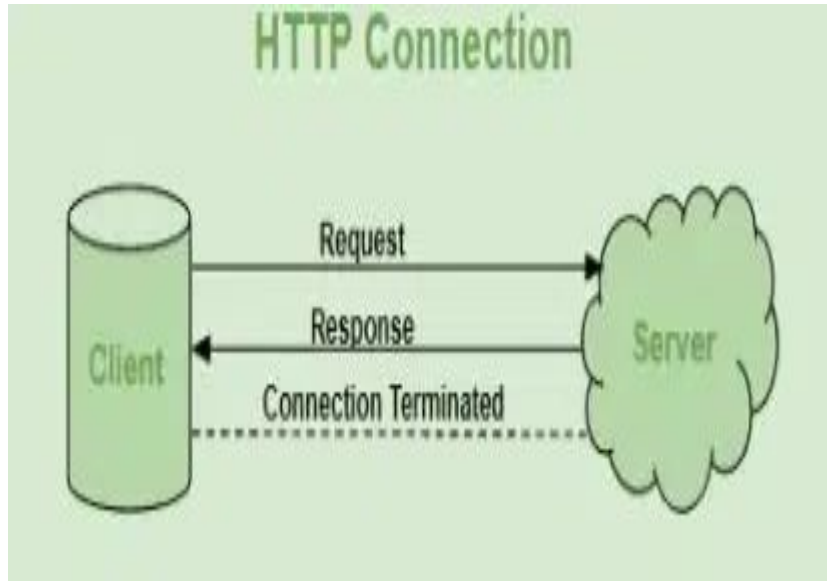
5. JavaScript Interpreter: As the name suggests, JavaScript Interpreter translates and executes the JavaScript code, which is included in a website. The translated results are sent to the rendering engine to display results on the device screen.

6. UI Backend: It is used to draw basic combo boxes and Windows (widgets). It specifies a generic interface, which is not platform-specific.

7. Data Storage: The data storage is a persistence layer that is used by the browser to store all sorts of information locally, like cookies. A browser also supports different storage mechanisms such as IndexedDB, WebSQL, localStorage, and FileSystem. It is a database stored on the local drive of your computer where the browser is installed. It handles user data like cache, bookmarks, cookies, and preferences.

What is HTTP ?

- **HTTP (Hypertext Transfer Protocol) is a fundamental protocol** of the Internet, enabling the transfer of data between a client and a server. It is the foundation of data communication for the World Wide Web.
- HTTP provides a standard between a web browser and a web server to establish communication. It is a set of rules for transferring data from one computer to another. Data such as text, images, and other multimedia files are shared on the World Wide Web. Whenever a web user opens their web browser, the user indirectly uses HTTP. It is an application protocol that is used for distributed, collaborative, hypermedia information systems.


HTTP Connection
Request
Response
Connection Terminated
Client
Server

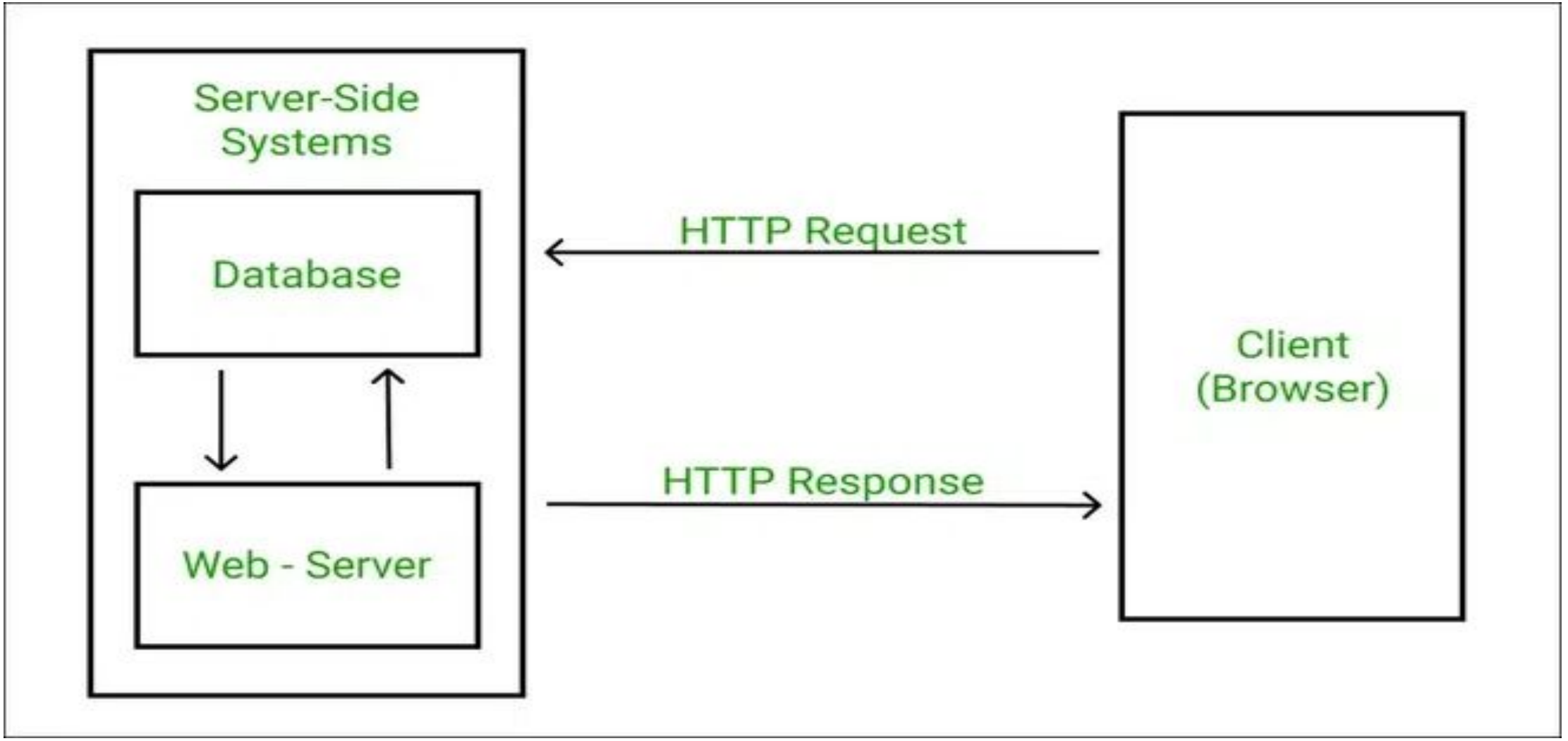
History of HTTP

- Tim Berners-Lee and his team at CERN are indeed credited with inventing the original HTTP protocol.
- HTTP version 0.9 was the initial version introduced in 1991.
- HTTP version 1.0 followed in 1996 with the introduction of RFC 1945.
- HTTP version 1.1 was introduced in January 1997 with RFC 2068, later refined in RFC 2616 in June 1999.
- HTTP version 2.0 was specified in RFC 7540 and published on May 14, 2015.
- HTTP version 3.0, also known as HTTP/3, is based on the QUIC protocol and is designed to improve web performance. It was renamed as Hyper-Text Transfer Protocol QUIC (HTTP/3) and developed by Google.

Features of HTTP:

- **Connectionless protocol:** HTTP is a connectionless protocol. HTTP client initiates a request and waits for a response from the server. When the server receives the request, the server processes the request and sends back the response to the HTTP client after which the client disconnects the connection. The connection between client and server exist only during the current request and response time only.
- **Media independent:** HTTP protocol is a media independent as data can be sent as long as both the client and server know how to handle the data content. It is required for both the client and server to specify the content type in MIME-type header.
- **Stateless:** HTTP is a stateless protocol as both the client and server know each other only during the current request. Due to this nature of the protocol, both the client and server do not retain the

HTTP Request/Response:



- HTTP is a request-response protocol, which means that for every request sent by a client (typically a web browser), the server responds with a corresponding response. The basic flow of an HTTP request-response cycle is as follows:
- **Client sends an HTTP request**: The client (usually a web browser) initiates the process by sending an HTTP request to the server. This request includes a request method (GET, POST, PUT, DELETE, etc.), the target URI (Uniform Resource Identifier, e.g., a URL), headers, and an optional request body.
- **Server processes the request**: The server receives the request and processes it based on the requested method and resource. This may involve retrieving data from a database, executing server-side scripts, or performing other operations.
- **Server sends an HTTP response:** After processing the request, the server sends an HTTP response back to the client. The response includes a status code (e.g., 200 OK, 404 Not Found), response headers, and an optional response body containing the requested data or content.
- **Client processes the response**: The client receives the server's response and processes it accordingly. For example, if the response contains an HTML page, the browser will render and display it. If it's an image or other media file, the browser will display or handle it appropriately.

- **Advantages**
- **Platform independence**: Works on any operating system
- **Compatibility**: Compatible with various protocols and technologies
- **Efficiency**: Optimized for performance
- **Security**: Supports encryption for secure data transfer

- **Disadvantages**
- **Lack of security**: Vulnerable to attacks like man in the middle
- **Performance issues:** Can be slow for large data transfers
- **Statelessness**: Requires additional mechanisms for maintaining state
-

HTTP Request:

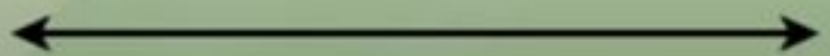
An HTTP request is made by a client (typically a web browser) to a server, requesting a particular action or resource. It consists of the following components:

- Request Line
  GET /api/users HTTP/1.1
- Headers
- Body

HTTP Response:

An HTTP response is sent by the server in response to an HTTP request. It contains the requested resource or provides information about the success or failure of the request. It consists of the following components:

- Status Line
  HTTP/1.1 200 OK
- Headers
- Body



HTTP Method:

GET method is used to request data from a server.
Imagine you're browsing the internet and you want to see a webpage. You type the URL into your browser, press Enter, and the webpage appears. In this case, you're using the GET method. It retrieves (gets) information from a server.

HTTP Method:

POST Method is used to send data to a server to create/update a resource. Let's say you're filling out a form on a website to create a new account. You enter your information, click the "Submit" button, and the website saves your details. This is an example of using the POST method. It sends data to the server for processing or creating something new.

HTTP Method:

PUT Method is also used to send data to a server to create/update a resource. Imagine you have an online profile, and you want to update your information, like your address or phone number. You make changes and click the "Save" button. By doing this, you're using the PUT method. It uploads (puts) the updated data to the server, replacing the existing information.

HTTP Method:

The DELETE method deletes the specified resource.
Suppose you have a file or an item you want to remove. You select it and click the "Delete" button. This action uses the DELETE method. It asks the server to delete the specified resource.

HTTP Method:

HEAD requests are useful for checking what a GET request will return before actually making a GET request - like before downloading a large file or response body.In response we can expect the size of the actual file.

HTTP Method:

The PATCH method is used to apply partial modifications to a resource.
Let's say you have an existing document, and you want to make a small change, like fixing a typo or updating a single field. Instead of replacing the entire document, you send a request with the PATCH method. It applies partial modifications to the resource.

HTTP Method:

The OPTIONS method describes the communication options for the target resource.

HTTP Method:

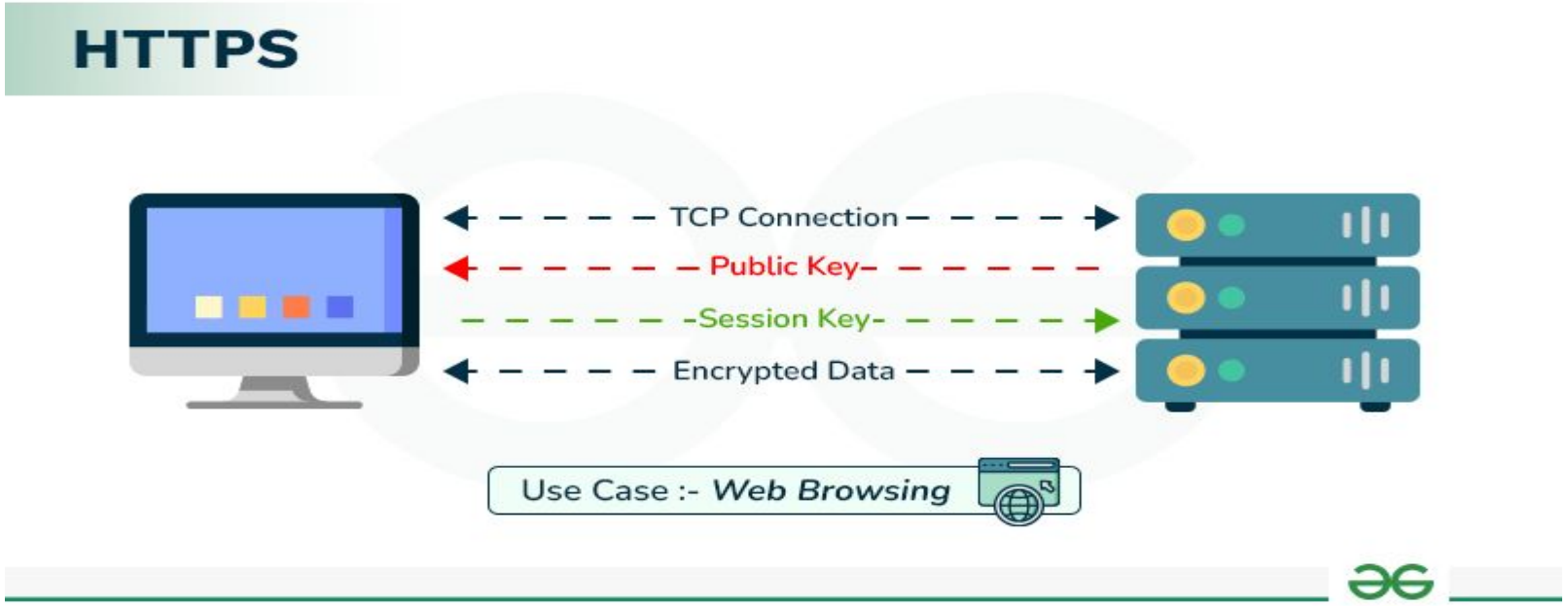
When a client wants to establish a secure connection with a server through an HTTP proxy, it sends an HTTP CONNECT request to the proxy server.

Methods of HTTP

HTTP Request	Description
GET	Asks to get the resource at the requested URL.
POST	Asks the server to accept the body info attached. It is like GET request with extra info sent with the request.
HEAD	Asks for only the header part of whatever a GET would return. Just like GET but with no body.
TRACE	Asks for the loopback of the request message, for testing or troubleshooting.
PUT	Says to put the enclosed info (the body) at the requested URL.
DELETE	Says to delete the resource at the requested URL.
OPTIONS	Asks for a list of the HTTP methods to which the thing at the request URL can respond

What is Hypertext Transfer Protocol Secure?

- Hypertext Transfer Protocol Secure is a protocol that is used to communicate between the user browser and the website. It also helps in the transfer of data. It is the secure variant of HTTP. To make the data transfer more secure, it is encrypted. Encryption is required to ensure security while transmitting sensitive information like passwords, contact information, etc.



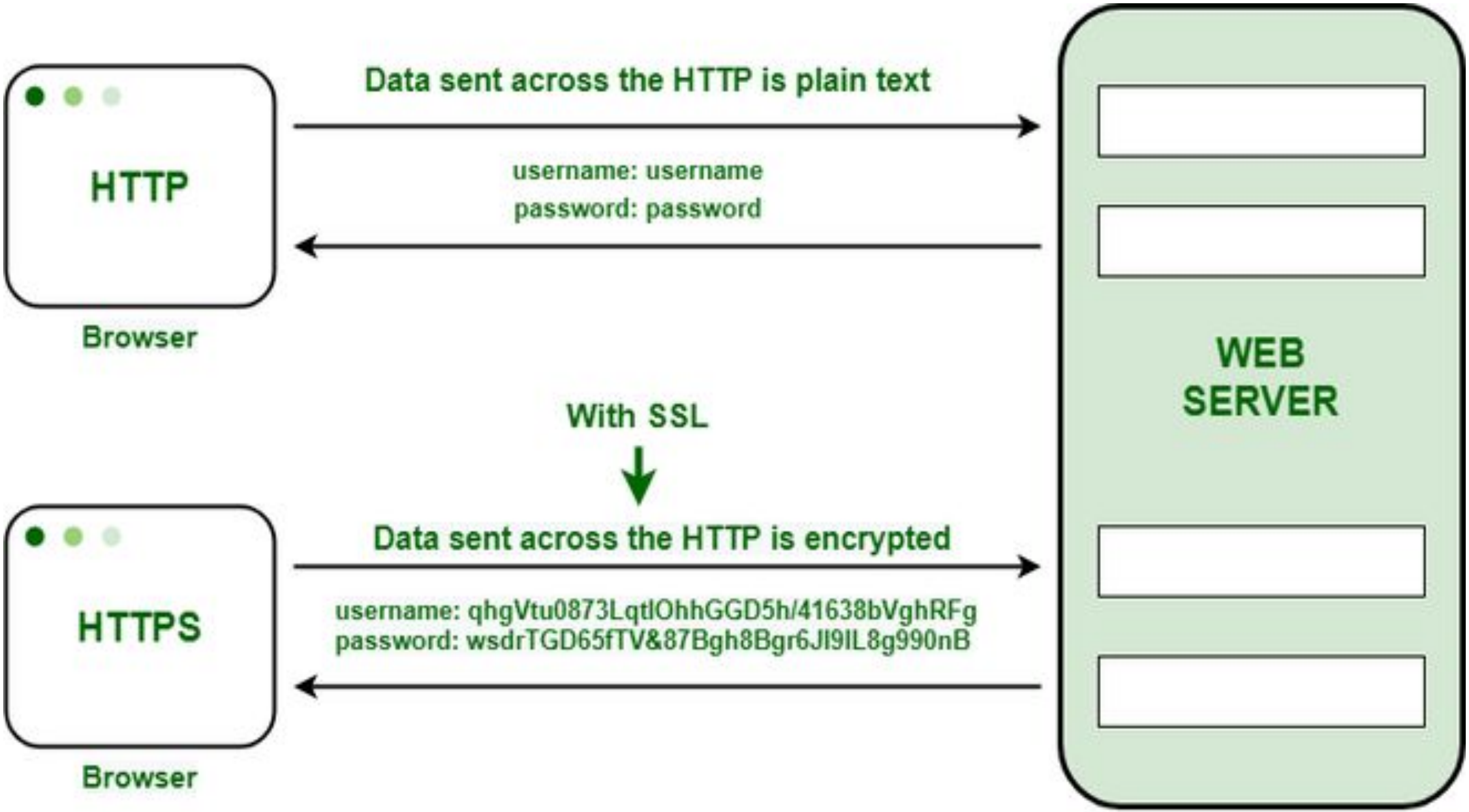
History of HTTPS

- 1994- Netscape Communications developed HTTPS for its Netscape Navigator web browser
- Earlier HTTPS used SSL protocol
- Secure Sockets Layer evolved to Transport Layer Security
- May 2000- RFC 2818 formally specified HTTPS
- Feb 2018- Google announcement about Chrome browser plan to mark HTTP sites as “not secure” after July 2018
- An effort to make Word Wide Web more secure
- **Purpose of HTTPS**
- HTTPS provides the **CONFIDENTIALITY** and **INTEGRITY** of the data between users computer and the website.
- HTTPS encrypt URL username , password , and sensitive information of users.

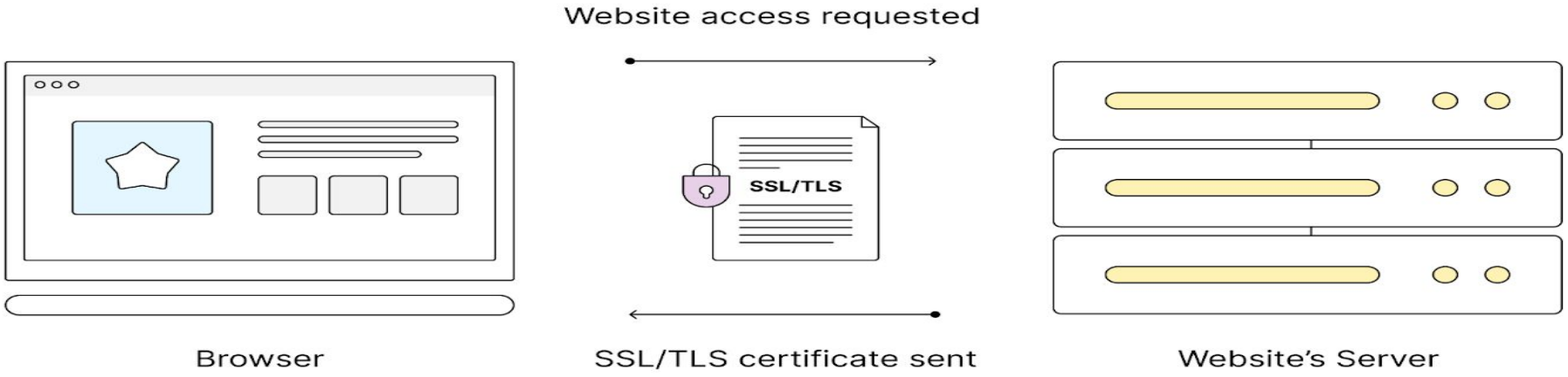
How Does HTTPS Work?

- HTTPS establishes the communication between the browser and the web server. It uses the Secure Socket Layer (SSL) and Transport Layer Security (TLS) protocol for establishing communication. The new version of SSL is TLS(Transport Layer Security).

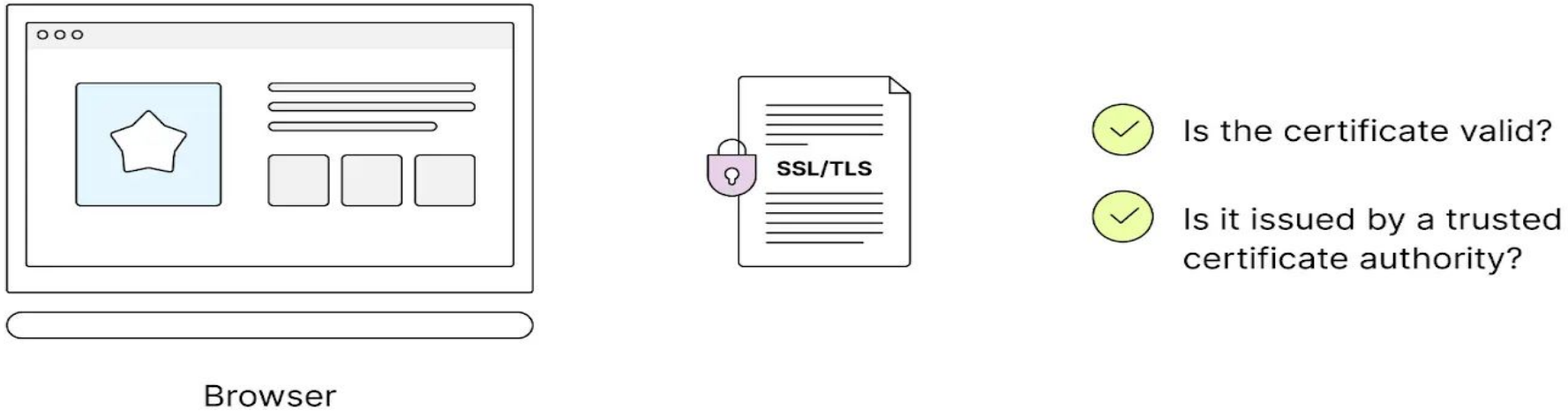
- HTTPS uses the conventional HTTP protocol and adds a layer of SSL/TLS over it. The workflow of HTTP and HTTPS remains the same, the browsers and servers still communicate with each other using the HTTP protocol. However, this is done over a secure SSL connection. The SSL connection is responsible for the encryption and decryption of the data that is being exchanged to ensure data safety.


HTTP
Browser
Data sent across the HTTP is plain text
username: username
password: password
WEB
SERVER
With SSL
HTTPS
Browser
Data sent across the HTTP is encrypted
username: qhgVtu0873LqtlOhhGGD5h/41638bVghRFg
password: wsdrTGD65fTV&87Bgh8Bgr6JI9IL8g990nB

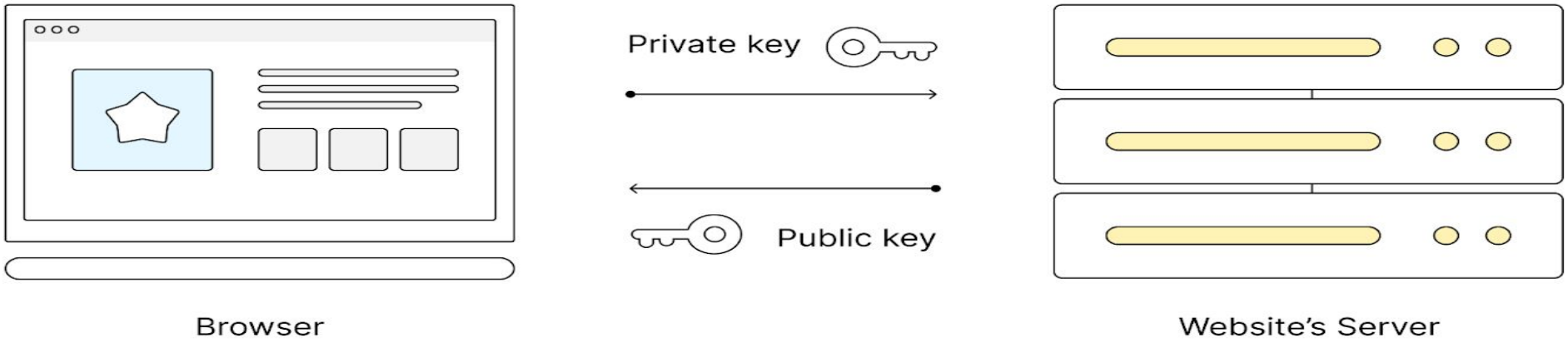
- 1. **Browser contacts website**: The user's web browser attempts to connect to a website using HTTPS
- 2. **SSL certificate sends**: The website's server responds by sending its SSL/TLS certificate to the browser. This certificate contains the website’s public key (encryption key) and is used to establish a secure connection.



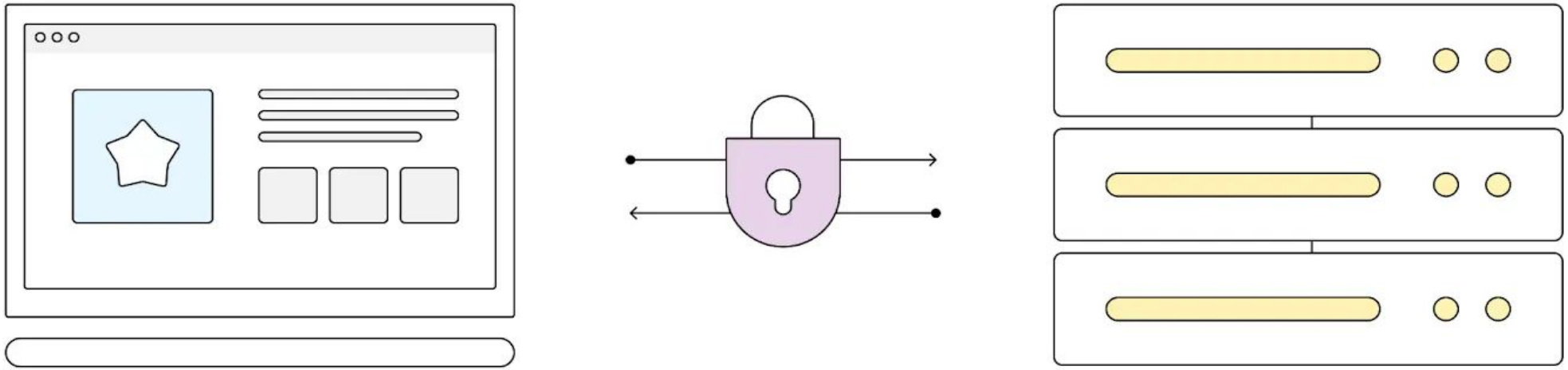
- **3. Browser verifies certificate**: The browser checks the certificate to ensure it’s valid and is issued by a trusted certificate authority (like GoDaddy, DigiCert, Comodo, etc.). This step is crucial for confirming a website’s authenticity.



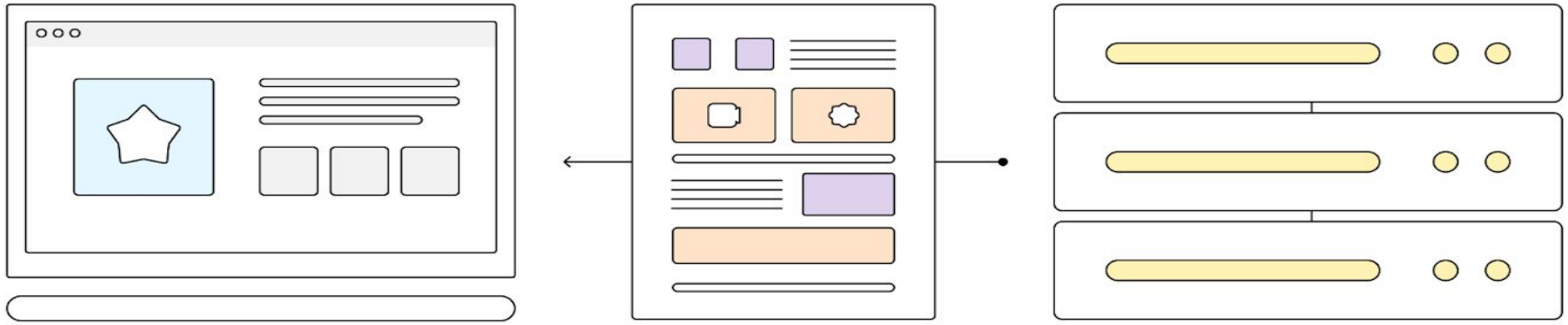
- **4. Encryption key exchange**: The browser and the server establish an encrypted connection by exchanging keys once the certificate is verified. The browser uses the server's public key to encrypt information, which can only be decrypted by the private key (i.e., the decryption key) the server holds



- 5. **Encrypted data transfer**: All data transferred between the browser and the server is encrypted after the secure connection is established. Which ensures It can’t be read by anyone intercepting the data.



- 6**. Data decryption and display**: The server decrypts the received data using the private key, processes it, and sends back the requested information. This data is also encrypted. The browser then decrypts the incoming data and displays the website content to the user.



Advantage of HTTPS

- **Secure Communication**: HTTPS establishes a secure communication link between the communicating system by providing encryption during transmission.
- **Data Integrity**: By encrypting the data, HTTPS ensures data integrity. This implies that even if the data is compromised at any point, the hackers won't be able to read or modify the data being exchanged.
- **Privacy and Security**: HTTPS prevents attackers from accessing the data being exchanged passively, thereby protecting the privacy and security of the users.
- **Faster Performance**: TTPS encrypts the data and reduces its size. Smaller size accounts for faster data transmission in the case of HTTPS.

Difference between HTTP and HTTPS

HTTP	HTTPS
HTTP stands for HyperText Transfer Protocol. In HTTP, the URL begins with “http://”.	HTTPS stands for HyperText Transfer Protocol Secure. In HTTPS, the URL starts with “https://”.
HTTP uses port number 80 for communication.	HTTPS uses port number 443 for communication.
Hyper-text exchanged using HTTP goes as plain text i.e. anyone between the browser and server can read it relatively easily if one intercepts this exchange of data and due to which it is Insecure.	HTTPS is considered to be secure but at the cost of processing time because Web Server and Web Browser need to exchange encryption keys using Certificates before actual data can be transferred.
HTTP Works at the Application Layer.	HTTPS works at Transport Layer.

HTTP Works at the Application Layer.	HTTPS works at Transport Layer.
HTTP does not use encryption, which results in low security in comparison to HTTPS.	HTTPS uses Encryption which results in better security than HTTP.
HTTP speed is faster than HTTPS.	HTTPS speed is slower than HTTP.
HTTP does not use data hashtags to secure data.	HTTPS will have the data before sending it and returning it to its original state on the receiver side.
HTTP is used to transfer text, video, and images via web pages.	HTTPS is used to transfer data securely via a network.

What is a Secure Socket Layer?

- SSL, or Secure Sockets Layer, is an Internet security protocol that encrypts data to keep it safe. It was created by Netscape in 1995 to ensure privacy, **authentication**, and **data integrity** in online communications. SSL is the older version of what we now call TLS (Transport Layer Security).

- Websites using SSL/TLS have **“HTTPS**” in their URL instead of **“HTTP.”**

- **Versions of SSL**
- SSL 1 – Never released due to high insecurity
- SSL 2 – Released in 1995
- SSL 3 – Released in 1996
- TLS 1.0 – Released in 1999
- TLS 1.1 – Released in 2006
- TLS 1.2 – Released in 2008
- TLS 1.3 – Released in 2018

How does SSL work?

- **Encryption**: SSL encrypts data transmitted over the web, ensuring privacy. If someone intercepts the data, they will see only a jumble of characters that is nearly impossible to decode.
- **Authentication**: SSL starts an authentication process called a handshake between two devices to confirm their identities, making sure both parties are who they claim to be.
- **Data Integrity**: SSL digitally signs data to ensure it hasn’t been tampered with, verifying that the data received is exactly what was sent by the sender.

Why is SSL Important?

- Originally, data on the web was transmitted in plaintext, making it easy for anyone who intercepted the message to read it. For example, if someone logged into their email account, their username and password would travel across the Internet unprotected.

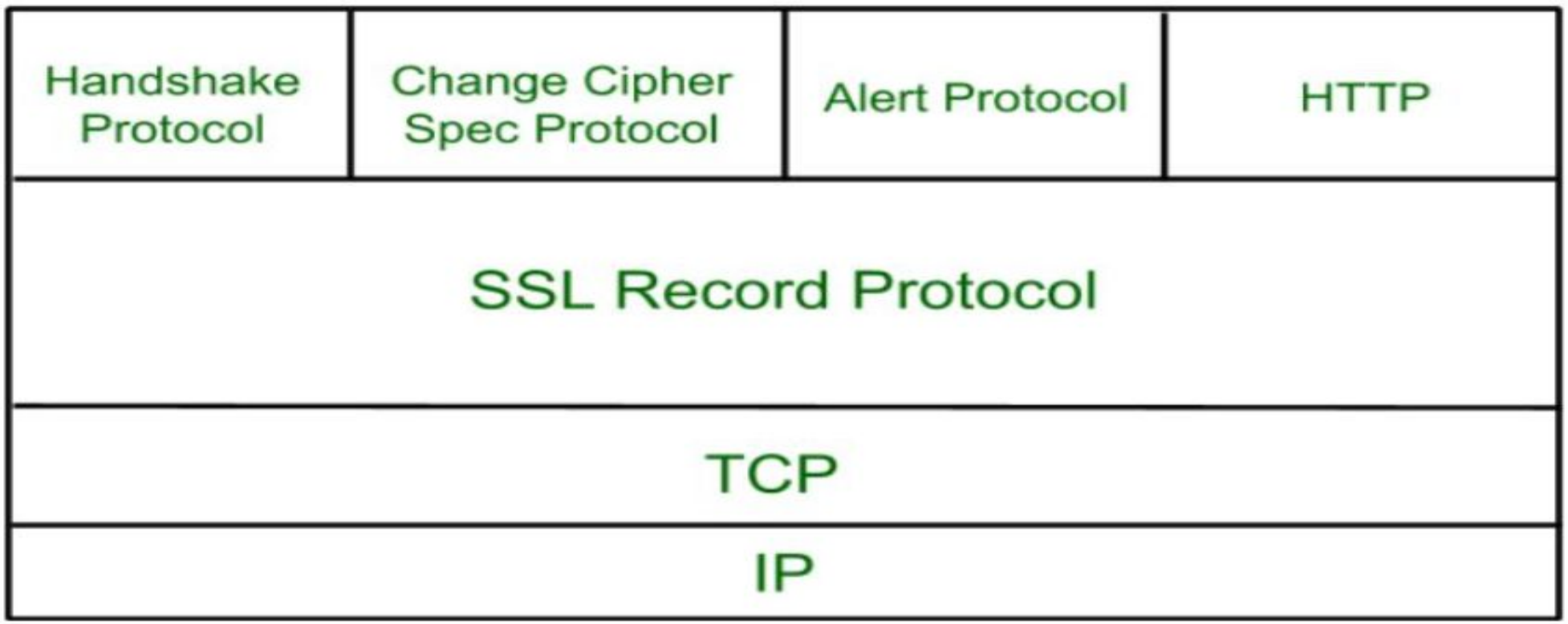
  SSL was created to solve this problem and protect user privacy. By encrypting data between a user and a web server, SSL ensures that anyone who intercepts the data sees only a scrambled mess of characters. This keeps the user’s login credentials safe, visible only to the email service

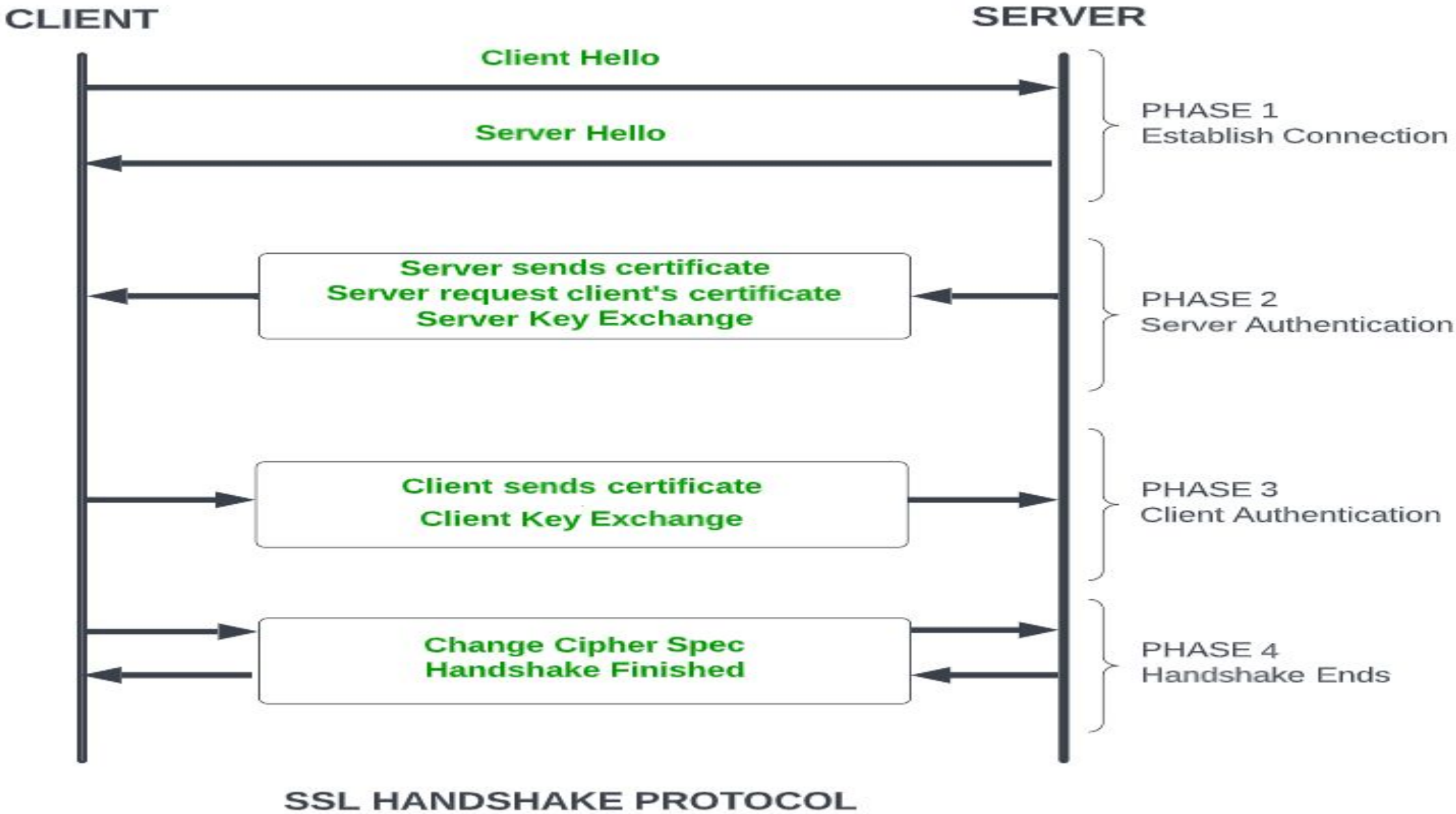
- **Additionally, SSL helps prevent cyber attacks by:**

- **Authenticating Web Servers**: Ensuring that users are connecting to the legitimate website, not a fake one set up by attackers.
- **Preventing Data Tampering**: Acting like a tamper-proof seal, SSL ensures that the data sent and received hasn’t been altered during transit.

Secure Socket Layer Protocols

- SSL Record Protocol
- Handshake Protocol
- Change-Cipher Spec Protocol
- Alert Protocol





SSL HANDSHAKE PROTOCOL

SSL Handshake Protocol

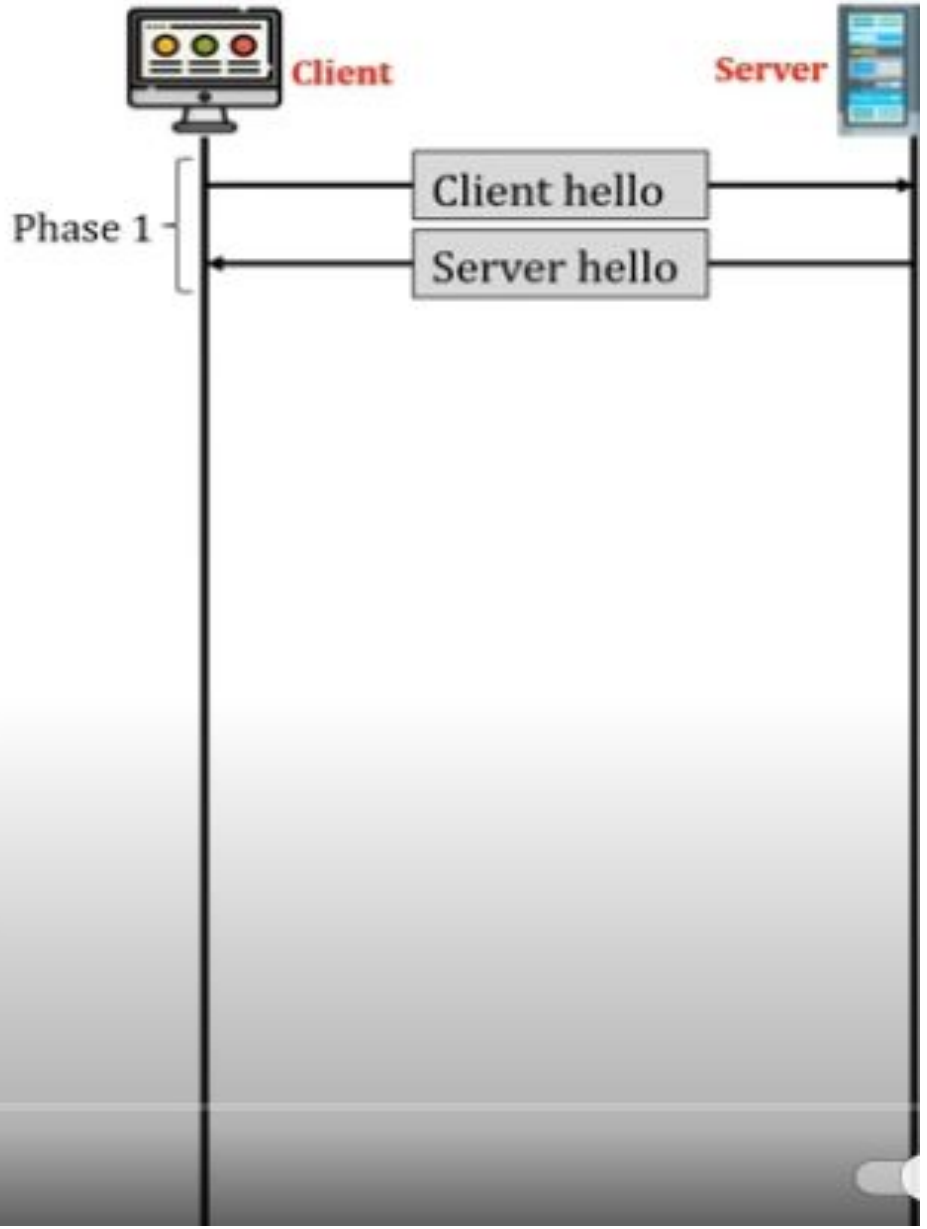
❑ Phase 1: Establishing security capabilities

1. Client Hello:

1. The highest SSL version number which the client can support.
2. A session ID that defines the session.
3. There is a cipher suite parameter that contains the entire cryptographic algorithm which supports client's system.
4. A list of compression methods that can be supported by client system.

2. Server Hello:

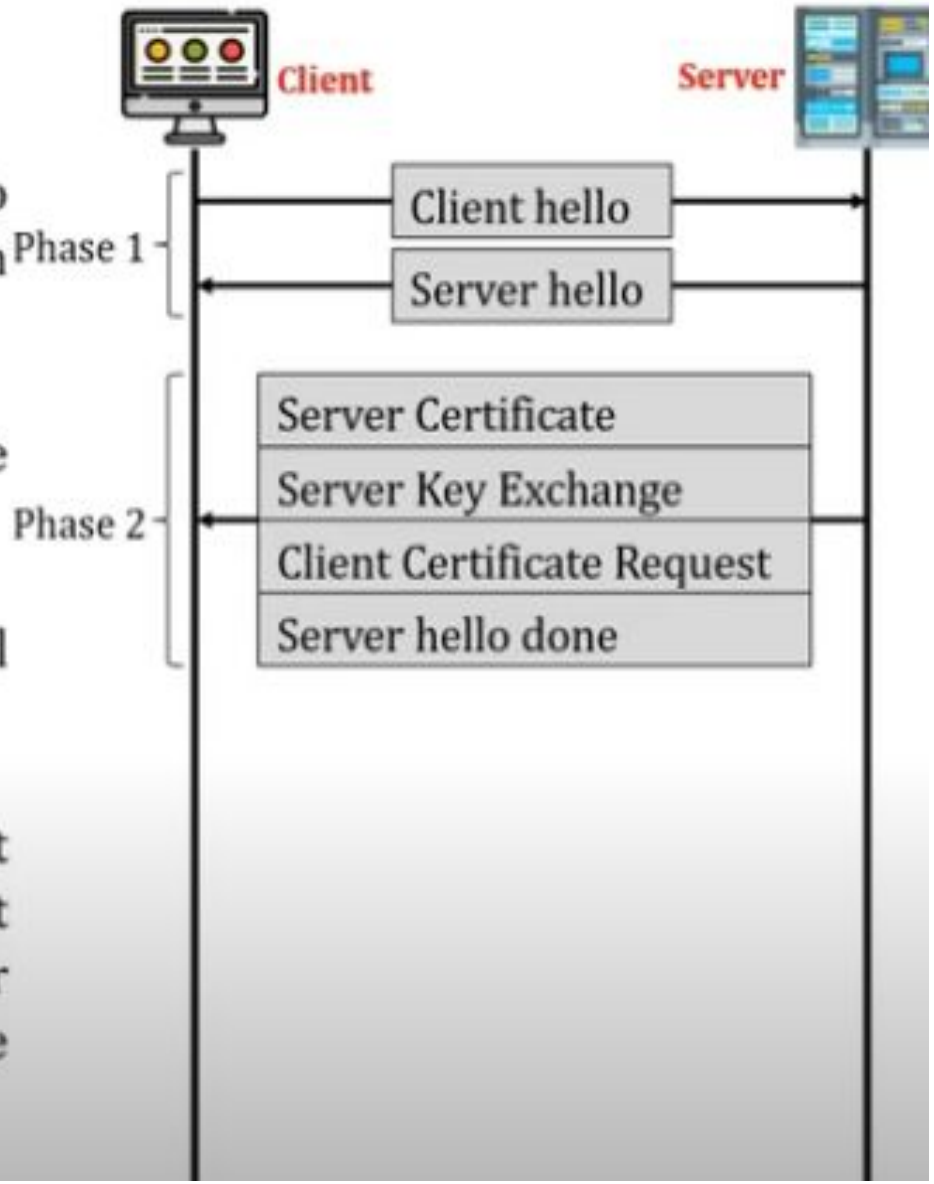
1. The highest SSL version number which the server can support.
2. A session ID that defines the session.
3. A cipher suite contains the list of all cryptographic algorithms that is sent by the client which the server will select the algorithm.
4. A list of compression method sent by the client from which the server will select the method.



SSL Handshake Protocol

❑ Phase 2: Server Authentication and Key Exchange

- **Certificate:** The server sends a certificate message to authentication itself to the client. If the key exchange algorithm is Diffie-Hellman than no need of authentication.
- **Server key exchange:** This is optional. It is used only if the server doesn't sends its digital certificate to client.
- **Certificate Request:** The server can request for the digital certificate of client. The client's authentication is optional.
- **Server Hello done:** The server message hello done is the last message in phase 2, this indicates to the client that the client can now verify all the certificates received by the server. After this hello message done, the server waits for the client side response in phase 3.



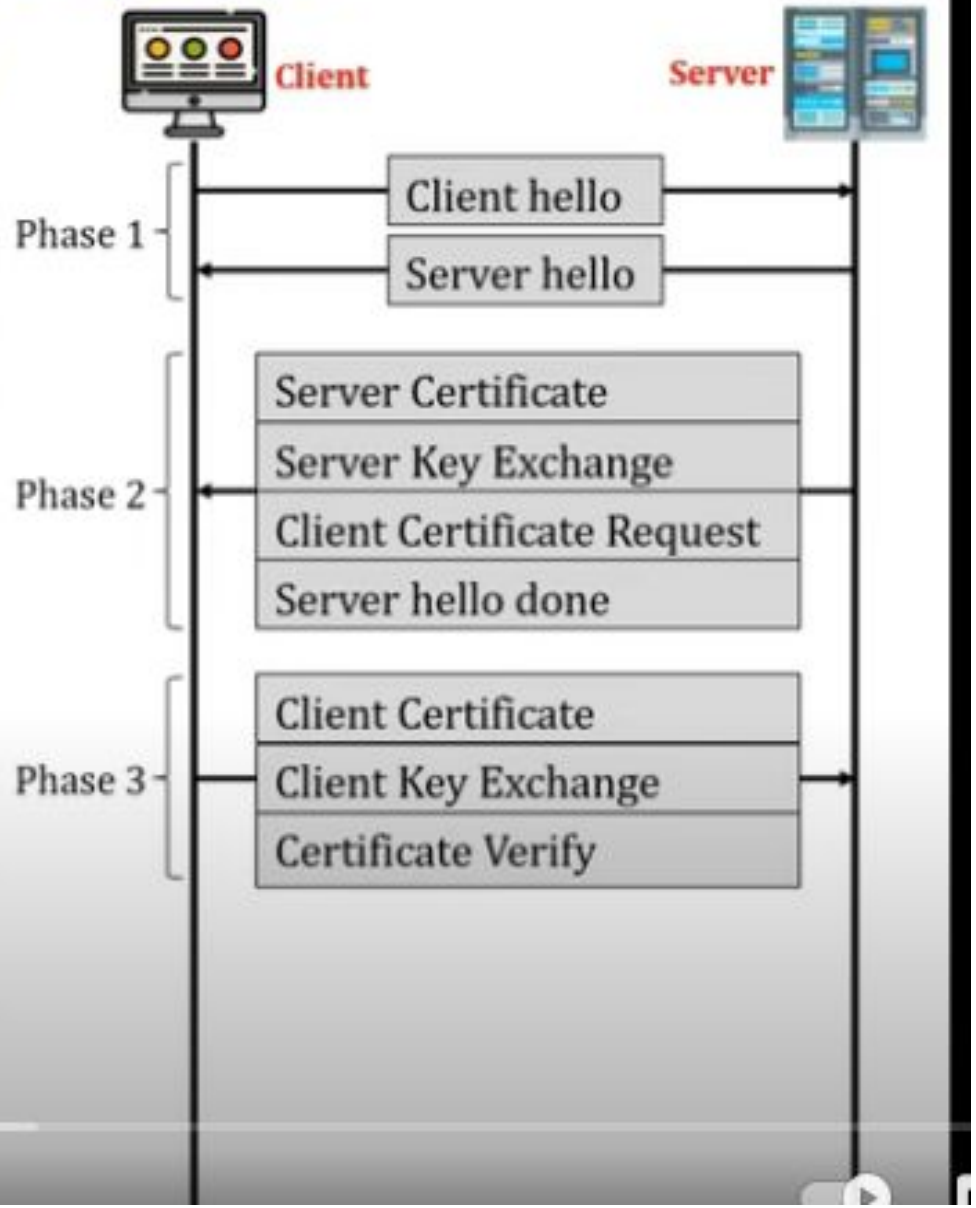
SSL Handshake Protocol

❑ Phase 3: Client Authentication and Key Exchange

1. **Client Certificate:** It is optional, it is only required if the server had requested for the client's digital certificate. If client doesn't have certificate it can be send no certificate message. Then it is upto server's decision whether to continue with the session or to abort the session.

2. **Client key exchange:** The client sends a client key exchange, the contents in this message are based on key exchange algorithms between both the parties.

3. **Certificate Verify:** It is necessary only if the server had asked for client authentication. The client has already sent its certificate to the server. Bit additionally if server wants then the client has to prove that it is authorized holder of the private key. The sever can verify the message with its public key already sent to ensure that the certificate belongs to client.



SSL Handshake Protocol

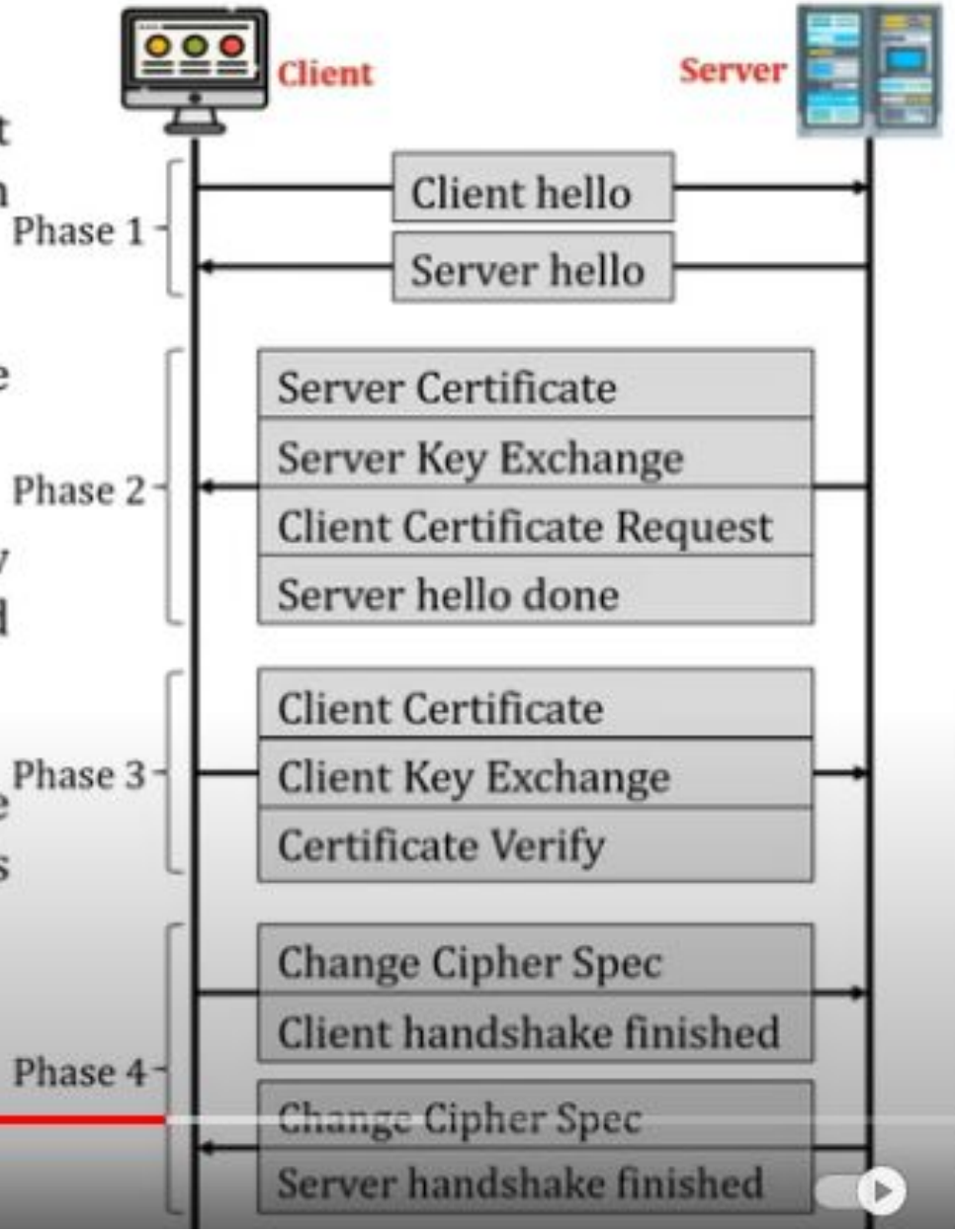
❑ Phase 4: Finish

1. **Change cipher spec:** It is a client side messages telling about the current status of cipher protocols and parameters which has been made active from pending state.

2. **Finished:** This message announce the finish of the handshaking protocol from client side.

3. **Change cipher spec:** This message is sent by server to show that it has made all the pending state of cipher protocols and parameters to active state.

4. **Finished:** This message announce the finish of the handshaking protocol from server and finally handshaking is totally completed.

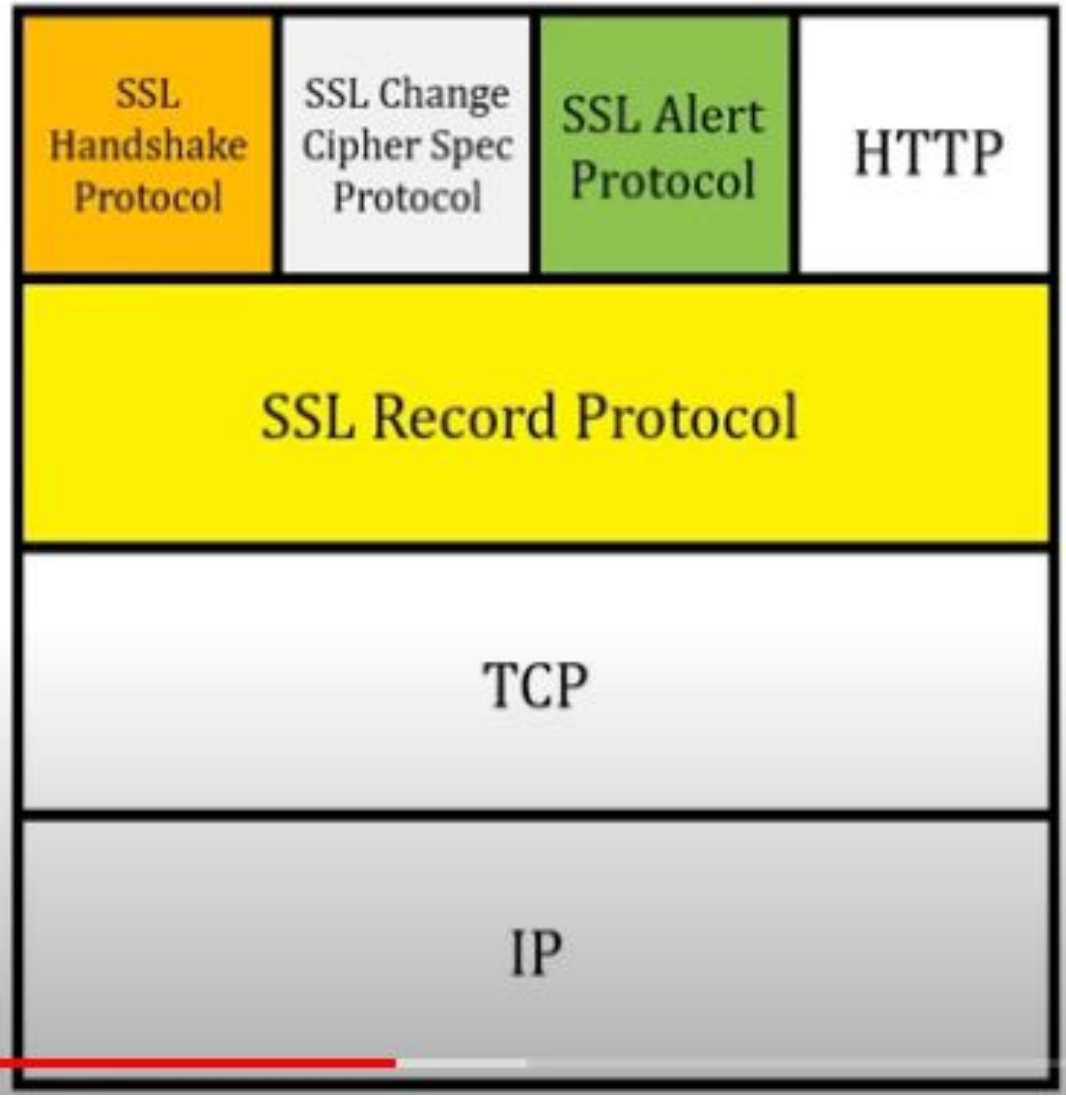


SSL Protocols (Explain with animation) ›

SSL Change Cipher Spec Protocol

- SSL Change Cipher Spec Protocol is upper layer protocol.
- It is the simplest protocol.

- This protocol consist of only single byte with value "1", as shown in figure.
- It consist of single message only.
- It copies pending state to current state, which updates the cipher suite to be used to this connection.



- **Alert Protocol**
- This protocol is used to convey SSL-related alerts to the peer entity. Each message in this protocol contains 2 bytes

The level is further classified into two parts:

Warning (level = 1)

This Alert has no impact on the connection between sender and receiver. Some of them are:

- **Bad Certificate:** When the received certificate is corrupt.
- **No Certificate:** When an appropriate certificate is not available.
- **Certificate Expired:** When a certificate has expired.
- **Certificate Unknown:** When some other unspecified issue arose in processing the certificate, rendering it unacceptable.
- **Close Notify**: It notifies that the sender will no longer send any messages in the connection.
- **Unsupported Certificate:** The type of certificate received is not supported.
- **Certificate Revoked:** The certificate received is in revocation list.

Fatal Error (level = 2):

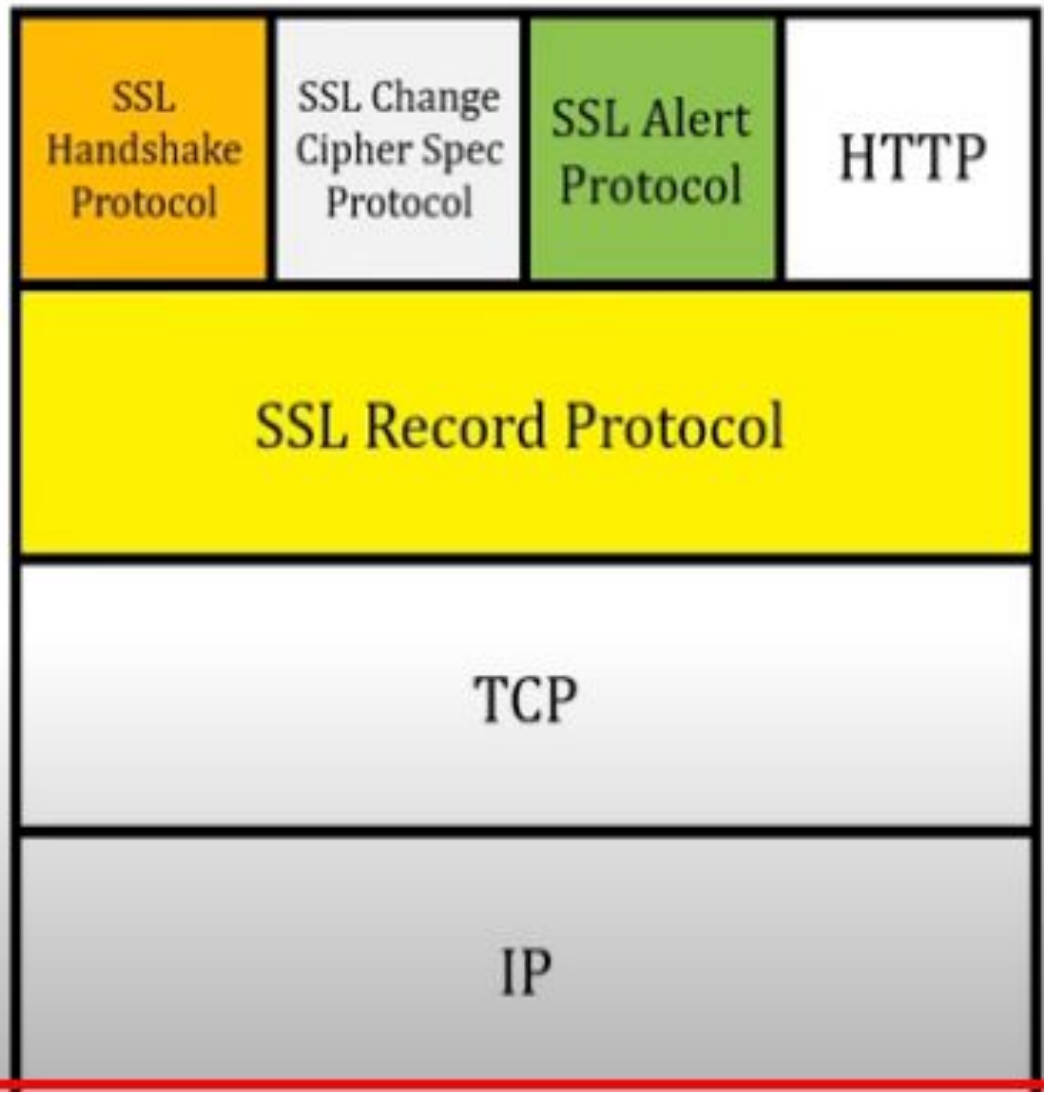
This Alert breaks the connection between sender and receiver. The connection will be stopped, cannot be resumed but can be restarted. Some of them are :

- **Handshake Failure:** When the sender is unable to negotiate an acceptable set of security parameters given the options available.
- **Decompression Failure**: When the decompression function receives improper input.
- **Illegal Parameters:** When a field is out of range or inconsistent with other fields.
- **Bad Record MAC:** When an incorrect MAC was received.
- **Unexpected Message:** When an inappropriate message is received.

The second byte in the Alert protocol describes the error.

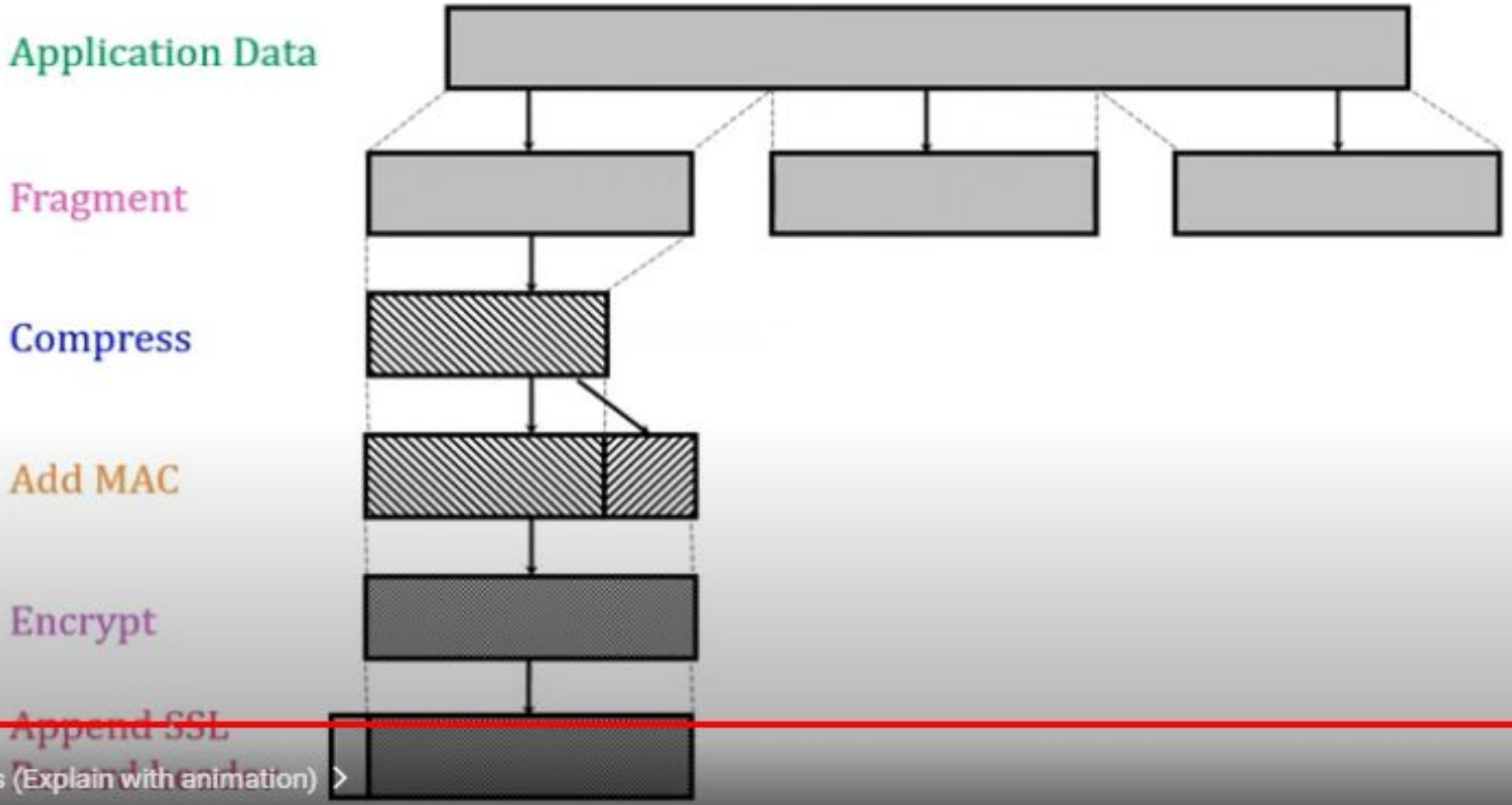
SSL Record Protocol

- SSL Record Protocol is second sub level protocol.
- It also called as lower level protocol.
- SSL Record Protocol provides following services:
  - Encrypted Data Transmission
  - Encapsulation of data
  - Data Confidentiality
  - Data Integrity



SSL Record Protocol

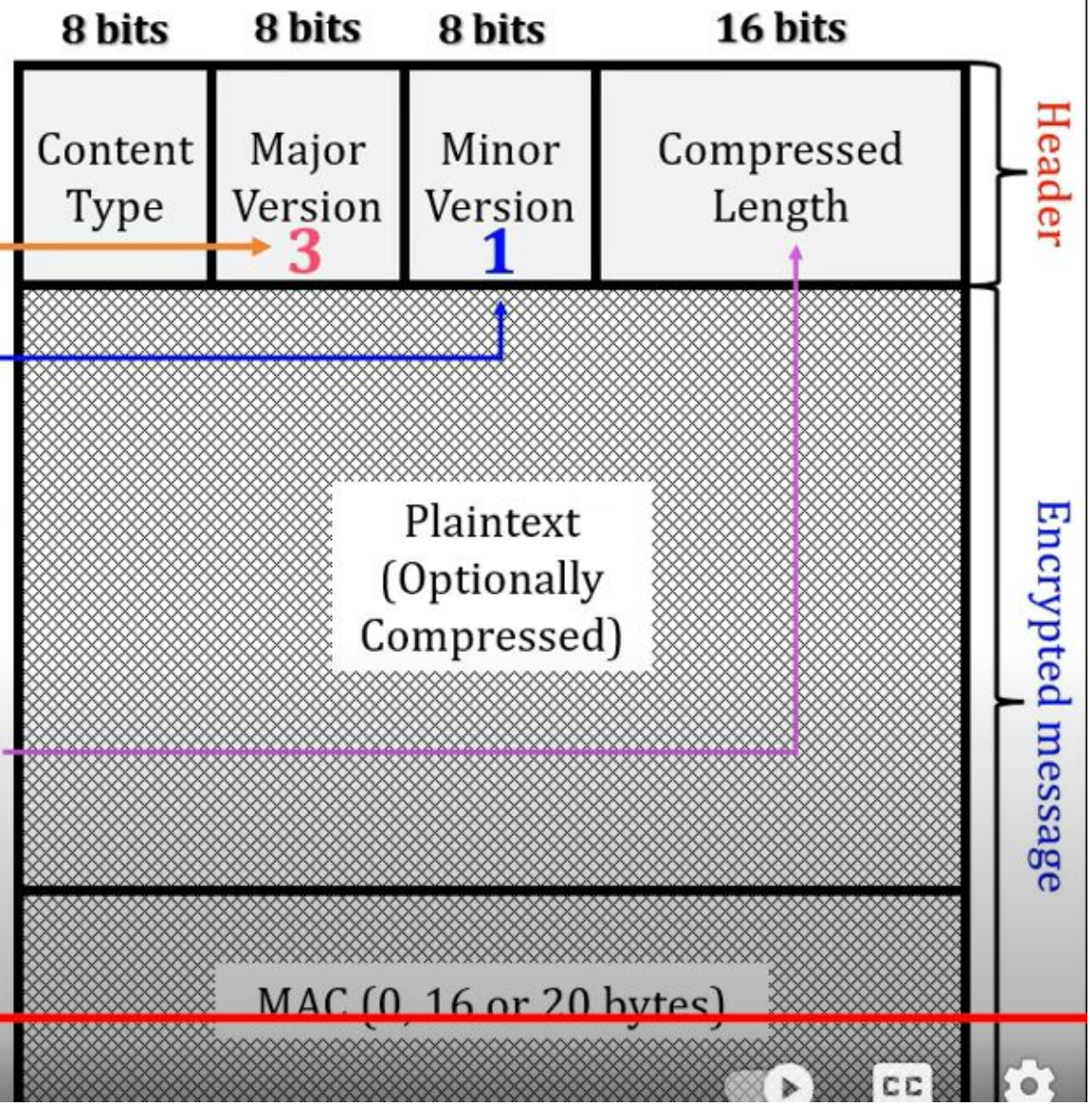
- How does SSL Record Protocol Work?

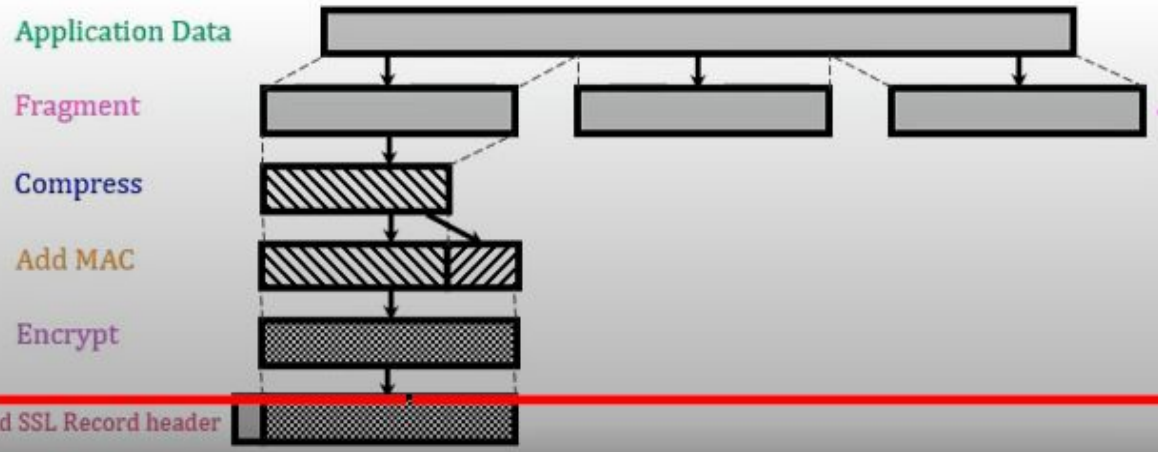


SSL Protocols (Explain with animation) >

SSL Record Protocol

- Following are components of SSL Record header format:
  - **Content Type:** Specify which protocol is used for processing.
  - **Major & Minor Version:** SSL 3.1
  - **Compressed length:** Length in bytes of the original plain text block.





- **Handshake Protocol**
- Handshake Protocol is used to establish sessions. This protocol allows the client and server to authenticate each other by sending a series of messages to each other. Handshake protocol uses four phases to complete its cycle.

- **Phase-1**: In Phase-1 both Client and Server send hello-packets to each other. In this IP session, cipher suite and protocol version are exchanged for security purposes.
- **Phase-2**: Server sends his certificate and Server-key-exchange. The server end phase-2 by sending the Server-hello-end packet.
- **Phase-3**: In this phase, Client replies to the server by sending his certificate and Client-exchange-key.
- **Phase-4**: In Phase-4 Change-cipher suite occurs and after this the Handshake Protocol ends.

Transport Layer Security (TLS)

- SSL stands for **Secure Socket Layer** while TLS stands for **Transport Layer Security**.
- Both Secure Socket Layer and Transport Layer Security are the protocols used to provide **security** between **web browsers and web servers**.
- The main difference between Secure Socket Layer and Transport Layer Security is that, in SSL (Secure Socket Layer), the **Message digest** is used to create a master secret and It provides the basic security services which are **Authentication** and **confidentiality**.
- while In TLS (Transport Layer Security), a **Pseudo-random function is used to create a master secret.**

HTTP over TLS

- TLS provides three primary services that help ensure the safety and security of data exchanged with it:

- **Authentication**
- Authentication lets each party to the communication verify that the other party is who they claim to be.

- **Encryption**
- Data is encrypted while being transmitted between the user agent and the server, in order to prevent it from being read and interpreted by unauthorized parties.

- **Integrity**
- TLS ensures that between encrypting, transmitting, and decrypting the data, no information is lost, damaged, tampered with, or falsified.

- A TLS connection starts with a handshake phase where a client and server agree on a shared secret and important parameters, like cipher suites, are negotiated. Once parameters and a data exchange mode where application data, such HTTP, is exchanged.

How Does SSL/TLS Encryption Work?

- SSL/TLS uses both asymmetric and symmetric encryption to protect the confidentiality and integrity of data-in-transit. Asymmetric encryption is used to establish a secure session between a client and a server, and symmetric encryption is used to exchange data within the secured session.
- A website must have an SSL/TLS certificate for their web server/domain name to use SSL/TLS encryption. Once installed, the certificate enables the **client and server to securely negotiate the level of encryption in the following steps:**

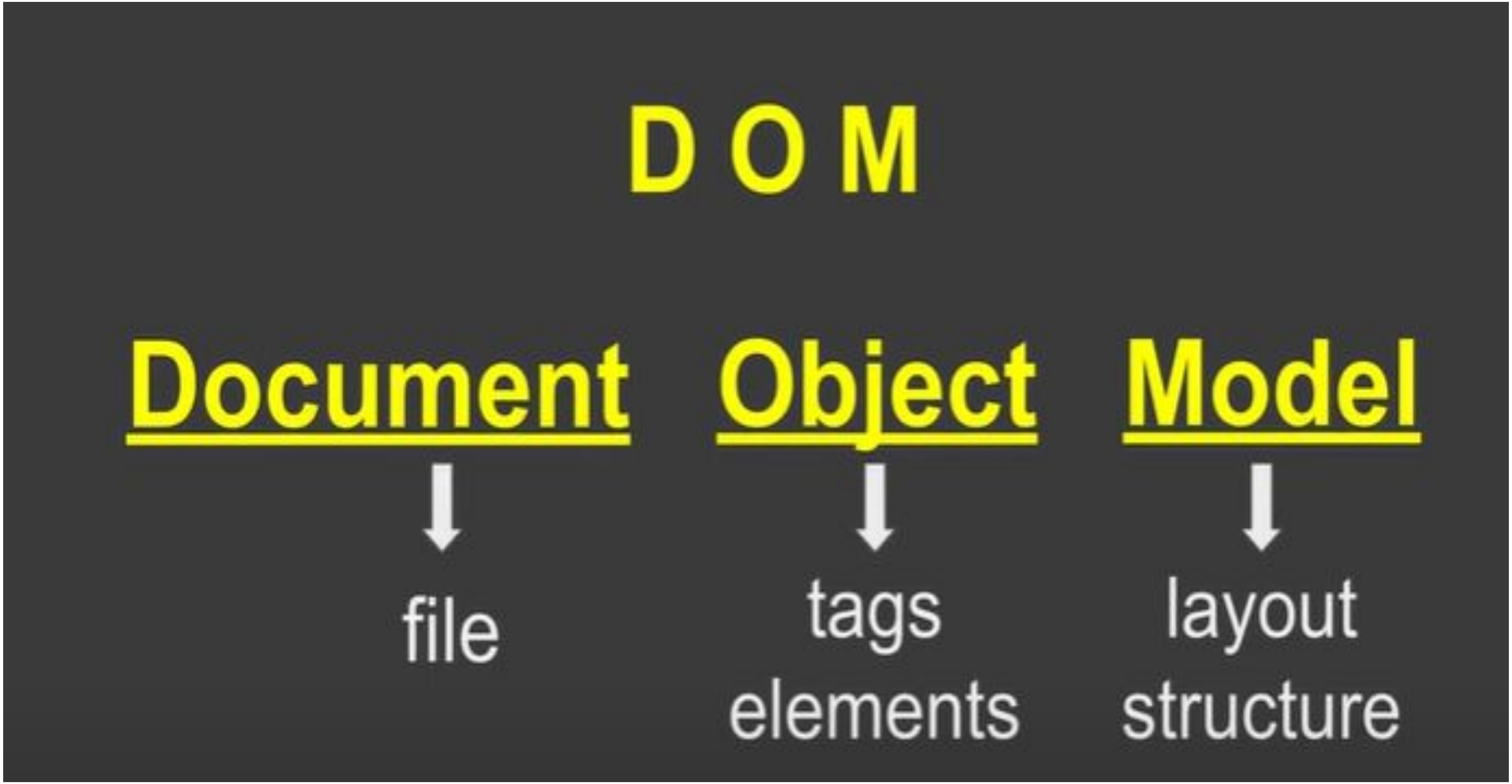
1. The client contacts the server using a secure URL (HTTPS…).
2. The server sends the client its certificate and public key.
3. The client verifies this with a Trusted Root Certification Authority to ensure the certificate is legitimate.
4. The client and server negotiate the strongest type of encryption that each can support.
5. The client encrypts a session (secret) key with the server's public key, and sends it back to the server.
6. The server decrypts the client communication with its private key, and the session is established.
7. The session key (symmetric encryption) is now used to encrypt and decrypt data transmitted between the client and server.

- Both the client and server are now using HTTPS (SSL/TLS + HTTP) for their communication. Web browsers validate this with a lock icon in the browser address bar. HTTPS functions over Port 443.
- Once you leave the website, those keys are discarded. On your next visit, a new handshake is negotiated, and a new set of keys are generated.

Differences between SSL and TLS

SSL	TLS
SSL stands for Secure Socket Layer.	TLS stands for Transport Layer Security.
SSL (Secure Socket Layer) is the 3.0 version.	TLS (Transport Layer Security) is the 1.0 version.
In SSL(Secure Socket Layer), the Message digest is used to create a master secret.	In TLS(Transport Layer Security), a Pseudo-random function is used to create a master secret.
In SSL(Secure Socket Layer), the Message Authentication Code protocol is used.	In TLS(Transport Layer Security), Hashed Message Authentication Code protocol is

SSL (Secure Socket Layer) is more complex than TLS(Transport Layer Security).	TLS (Transport Layer Security) is simple.
SSL (Secure Socket Layer) is less secured as compared to TLS(Transport Layer Security).	TLS (Transport Layer Security) provides high security.
SSL is less reliable and slower.	TLS is highly reliable and upgraded. It provides less latency.
SSL has been depreciated.	TLS is still widely used.
SSL uses port to set up explicit connection.	TLS uses protocol to set up implicit connection.


DOM
Document
Object
Model
file
tags
elements
layout
structure

HTML DOM (Document Object Model)

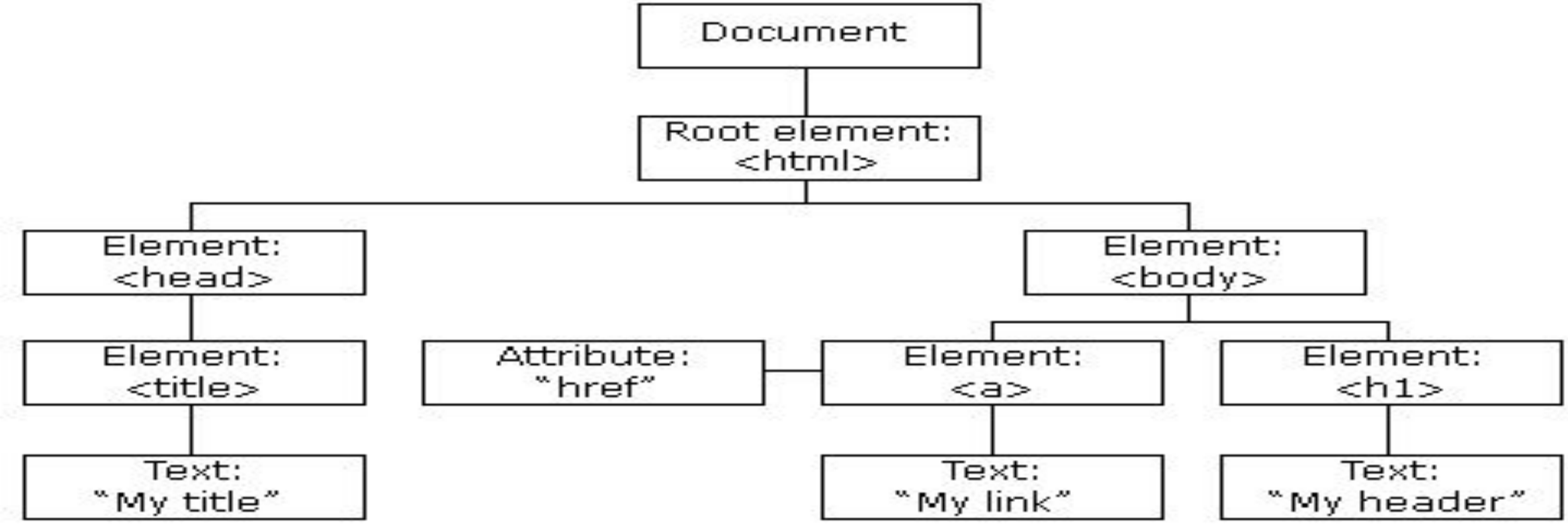
- **HTML DOM (Document Object Model)** is a hierarchical representation of HTML documents. It defines the **structure and properties** of elements on a webpage, enabling JavaScript to dynamically access, manipulate, and update content, enhancing interactivity and functionality.

- **DOM**, or **Document Object Model**, is a programming interface that represents structured documents like HTML and XML as a tree of objects. It defines how to access, manipulate, and modify document elements using scripting languages like JavaScript

- **Note**: It is called a Logical structure because DOM doesn't specify any relationship between objects.

The HTML DOM (Document Object Model)

- When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.
- The **HTML DOM** model is constructed as a tree of **Objects**:

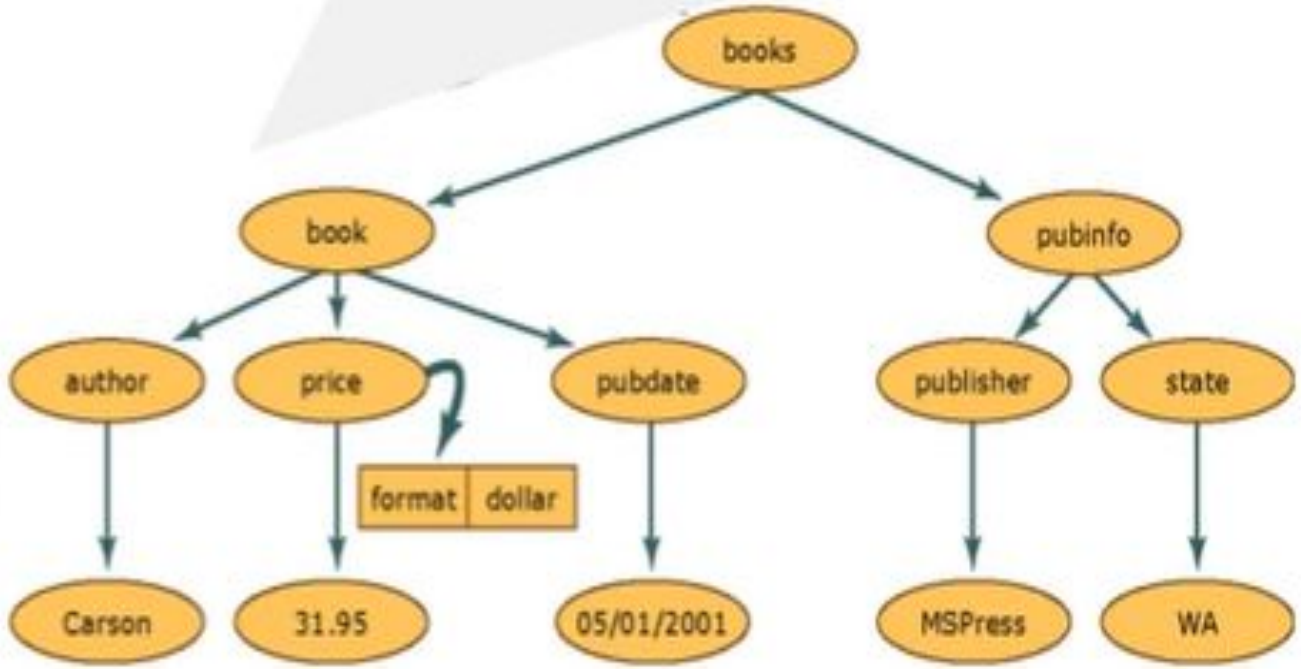


- **With the object model, JavaScript gets all the power it needs to create dynamic HTML**:

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

```
<books>
 <book>
    <author>Carson</author>
    <price format="dollar">31.95</price>
    <pubdate>05/01/2001</pubdate>
 </book>
 <pubinfo>
    <publisher>MSPress</publisher>
    <state>WA</state>
 </pubinfo>
</books>
```

DOM represents the content of xml or html document as tree structure



Can easily read, access, update the contents of the document

```xml
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>

 <book category="cooking">
  <title lang="en">Everyday Italian</title>
  <author>Giada De Laurentiis</author>
  <year>2005</year>
  <price>30.00</price>
 </book>

 <book category="children">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
 </book>
```

All XML elements can be accessed through the XML DOM

```
xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue;
```

This code retrieves the text value of the first <title> element in an XML document

Categories of DOM

- DOM Core – It specifies a generic model for viewing and manipulating a marked up document as a tree structure.
- DOM HTML – It specifies an extension to the core DOM for use with HTML. This represents DOM Level 0 with capabilities for manipulating all of the HTML element objects.
- DOM CSS – It specifies the interfaces necessary to manipulate CSS rules programmatically.
- DOM Events – It adds event handling to the DOM.
- DOM XML – It specifies an extension to the Core Dom for use with XML.

Why is DOM Required?

- HTML is used to **structure** the web pages and Javascript is used to add **behavior** to our web pages. **When an HTML file is loaded into the browser, the JavaScript can not understand the HTML document** directly. So **it interprets and interacts** with the Document Object Model (DOM), which is created by the browser based on the HTML document.
- **DOM is basically the representation of the same HTML document but in a tree-like structure composed of objects.** JavaScript can not understand the tags(<h1>H</h1>) in HTML document but can understand object h1 in DOM.
- JavaScript interprets DOM easily, using it as a bridge to access and manipulate the elements. DOM Javascript allow access to each of the objects (h1, p, etc) by using different functions.

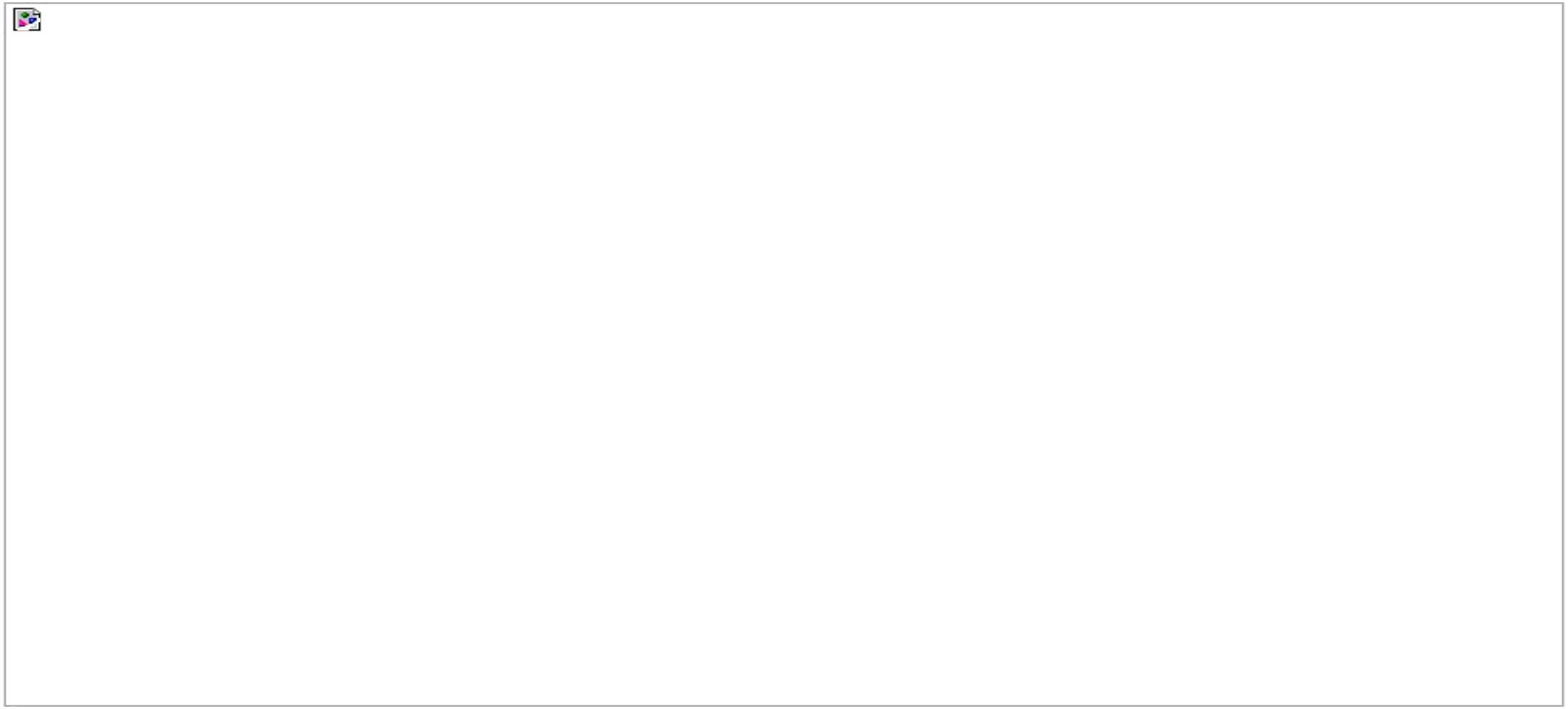
The Document Object Model (DOM) is essential in web development for several reasons:

- **Dynamic Web Pages:** It allows us to create dynamic web pages. It enables the JavaScript to access and manipulate page content, structure, and style dynamically which gives interactive and responsive web experiences, such as updating content without reloading the entire page or responding to user actions instantly.
- **Interactivity:** With the DOM, we can respond to user actions (like clicks, inputs, or scrolls) and modify the web page accordingly.
- **Content Updates:** When we want to update the content without refreshing the entire page, the DOM enables targeted changes making the web applications more efficient and user-friendly.
- **Cross-Browser Compatibility:** Different browsers may render HTML and CSS in different ways. The DOM provides a standardized way to interact with page elements.
- **Single-Page Applications (SPAs):** Applications built with frameworks such as React or Angular, heavily rely on the DOM for efficient rendering and updating of content within a single HTML page without reloading the full page.

Structure of DOM

- DOM can be thought of as a Tree or Forest (more than one tree). The term **structure model** is sometimes used to describe the tree-like representation of a document.
- Each branch of the tree ends in a node, and each node contains objects Event listeners can be added to nodes and triggered on an occurrence of a given event. One important property of DOM structure models is ***structural isomorphism***: if any two DOM implementations are used to create a representation of the same document, they will create the same structure model, with precisely the same objects and relationships.
- **Why DOM is called an Object Model?**
- Documents are modeled using objects, and the model includes not only the structure of a document but also the behavior of a document and the objects of which it is composed like tag elements with attributes in HTML.

Properties of DOM



- **Window Object:** Window Object is object of the browser which is always at top of the hierarchy. It is like an API that is used to set and access all the properties and methods of the browser. It is automatically created by the browser.
- **Document object:** When an HTML document is loaded into a window, it becomes a document object. The ‘document’ object has various properties that refer to other objects which allow access to and modification of the content of the web page. If there is a need to access any element in an HTML page, we always start with accessing the ‘document’ object. Document object is property of window object.
- **Form Object:** It is represented by ***form*** tags.
- **Link Object:** It is represented by ***link*** tags.
- **Anchor Object:** It is represented by ***a href*** tags.
- **Form Control Elements:** Form can have many control elements such as text fields, buttons, radio buttons, checkboxes, etc.

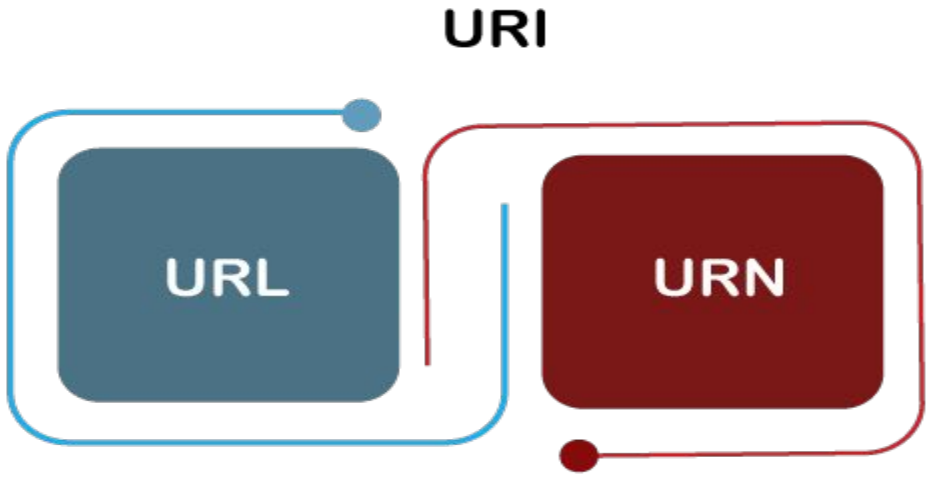
Methods of Document Object

- **DOM** provides various methods that allows users to interact with and manipulate the document. Some commonly used DOM methods are
- .

Method	Description
write("string")	Writes the given string on the document.
getElementById()	Returns the element having the given id value.
getElementsByName()	Returns all the elements having the given name value.
getElementsByTagName()	Returns all the elements having the given tag name.
getElementsByClassName()	Returns all the elements having the given class name.

Difference between URI and URL

- A **URI** or **Uniform Resource Identifier** is a string identifier that refers to a resource on the internet. It is a string of characters that is used to identify any resource on the internet using location, name, or both.
- A URI has two subsets; **URL (Uniform Resource Locator)** and **URN (Uniform Resource Number)**. If it contains only a name, it means it is not a URL. Instead of directly URI, we mostly see the URL and URN in the real world



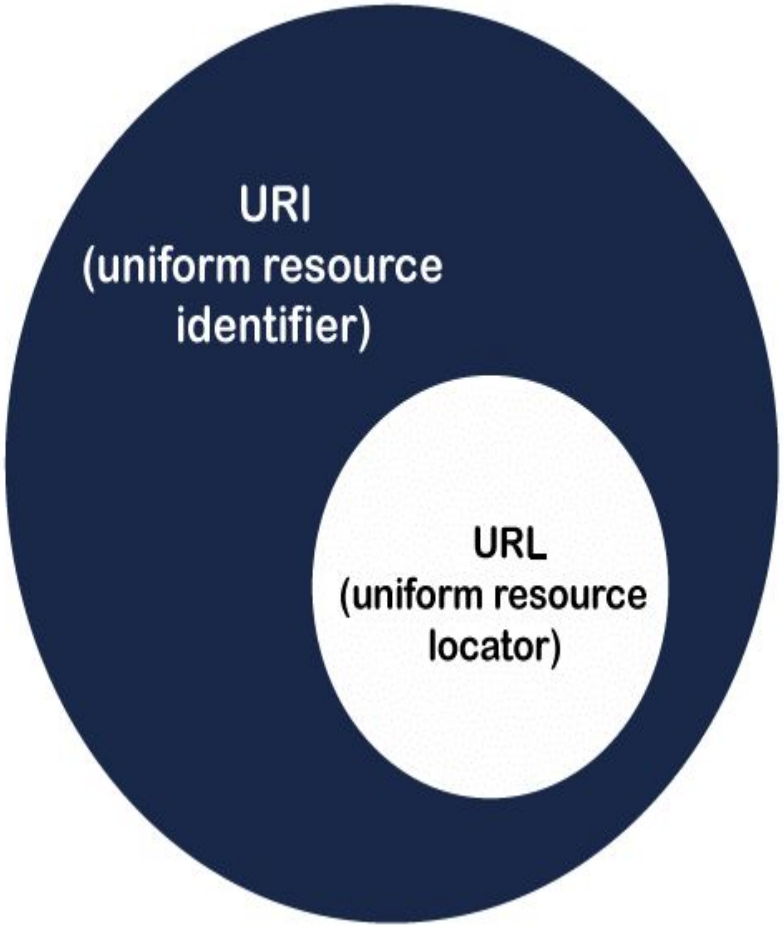
- A URI contains **scheme, authority, path, query, and a fragment.** Some most common URI schemes are **HTTP, HTTPs, ftp , telnet, etc**.

- **Syntax of URI**
- The Syntax of URI is given below:
- **scheme:[//authority]path[?query][#fragment]**
- **Some examples of URI**
- mailto:hey.Doe@example.com
- news:comp.infosystems.www.servers.unix
- urn:oasis:names:specification:docbook:dtd:xml:4.1.2
-

What is the URL?

- A URL or **Uniform Resource Locator** is used to find the location of the resource on the web. It is a reference for a resource and a way to access that resource. A URL always shows a unique resource, and it can be an HTML page, a CSS document, an image, etc.
- A URL uses a protocol for accessing the resource, which can be HTTP, HTTPS, FTP, etc.
- It is mainly referred to as the address of the website, which a user can find in their address bars.
- An example of an URL is given below:

- **Note: All URLs can be URIs, but all URIs cannot be URLs. It is because a URI contains both URL and URN and represent URL or URN, or both.**

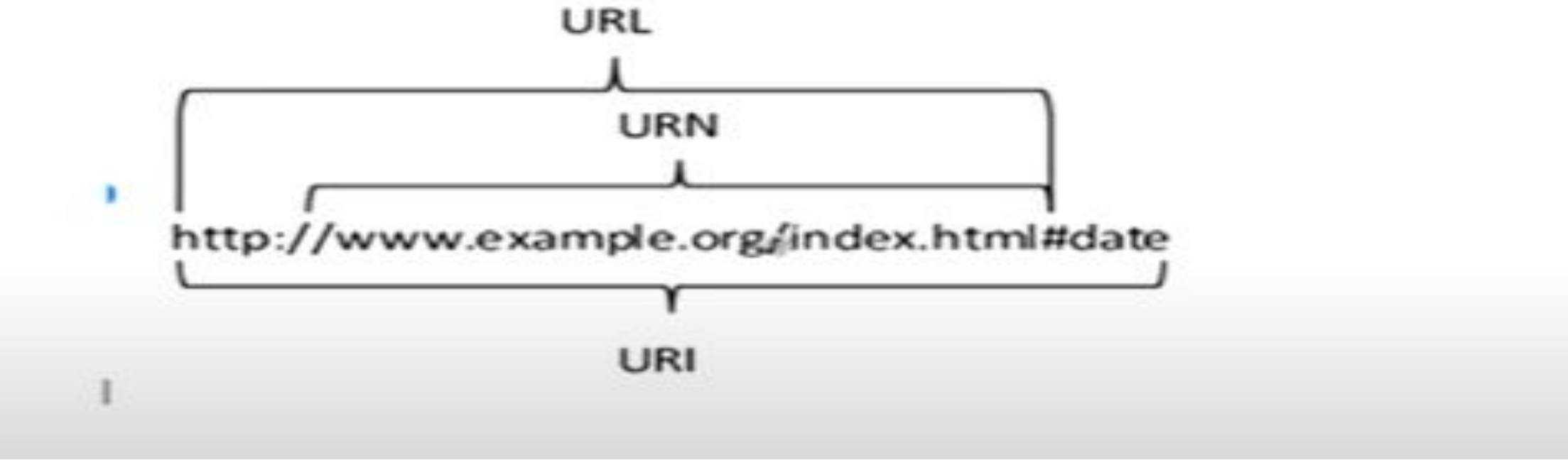


Syntax of URL

Each HTTP URL follow the syntax of its generic URI. Hence the syntax of the URL is also similar to the syntax of URI. It is given below:

scheme:[//authority]path[?query][#fragment]

- **URN stands for Universal Resource Name**. it refer the location of the content using anchor Identifier. URNs are URL but not vice Versa. There is a prefix URN for all the URNs. When you specify “access mechanism, location’’ And URI becomes URL.

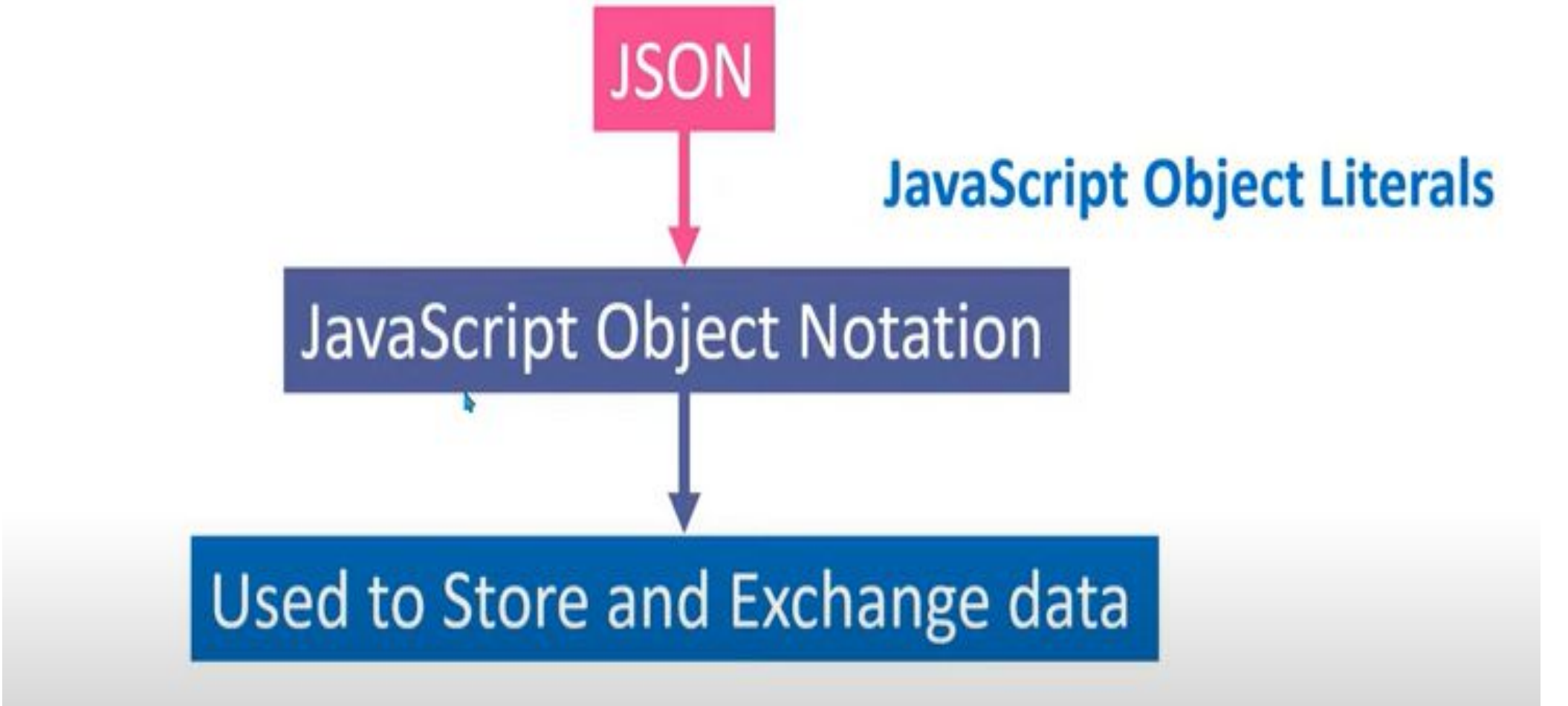


Difference chart between URI and URL

URI	URL
URI is an acronym for Uniform Resource Identifier.	URL is an acronym for Uniform Resource Locator.
URI contains two subsets, URN, which tell the name, and URL, which tells the location.	URL is the subset of URI, which tells the only location of the resource.
All URIs cannot be URLs, as they can tell either name or location.	All URLs are URIs, as every URL can only contain the location.
A URI aims to identify a resource and differentiate it from other resources by using the name of the resource or location of the resource.	A URL aims to find the location or address of a resource on the web.
An example of a URI can be ISBN 0-486-35557-4.	An example of an URL is https://www.javatpoint.com.

It is commonly used in XML and tag library files such as JSTL and XSTL to identify the resources and binaries.	It is mainly used to search the webpages on the internet.
The URI scheme can be protocol, designation, specification, or anything.	The scheme of URL is usually a protocol such as HTTP, HTTPS, FTP, etc.

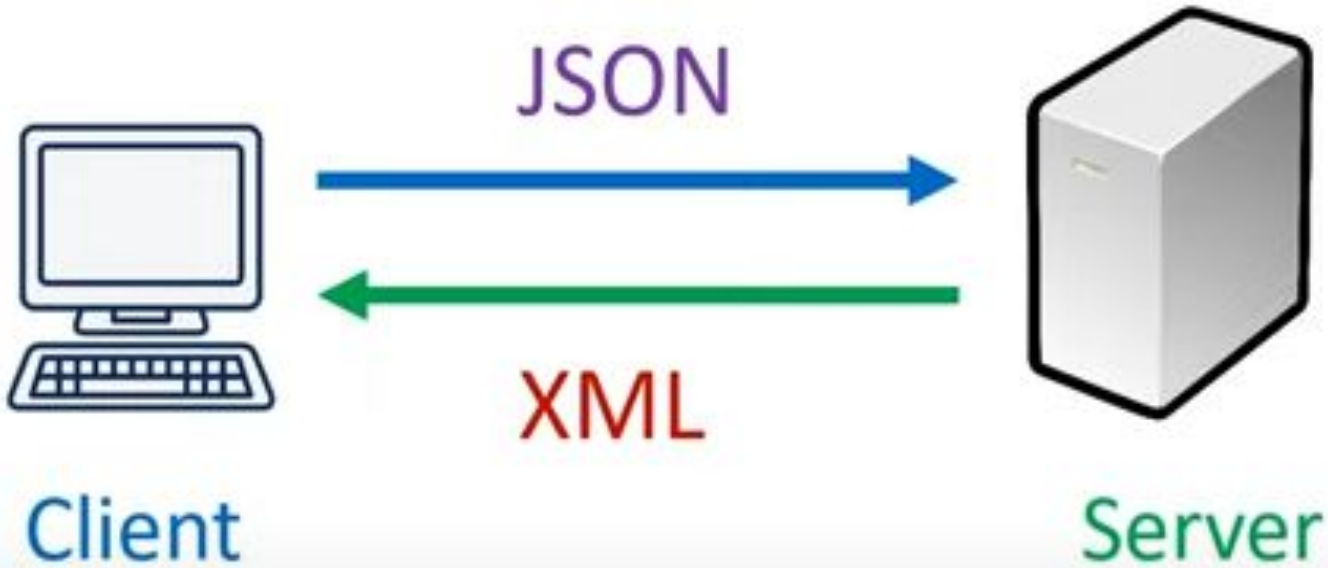
What is JSON ?


JSON
JavaScript Object Literals
JavaScript Object Notation
Used to Store and Exchange data

- **JSON** stands for **JavaScript Object Notation** is a lightweight and human-readable format for storing and exchanging data. It is a format for structuring data. This format is used by different web applications to communicate with each other. It has become the actual standard for data communication across web applications due to its simplicity and flexibility.
- **JSON**, short for **JavaScript Object Notation**, makes sharing data simple and straightforward. Created by Douglas Crockford, it's designed for easy reading and writing by humans, and easy parsing and generating by computers. Its main goal was to make a text format that's good at showing simple data like lists and text, and really useful for websites.
- **JSON** is special because it's very clear and easy to use, and it uses a ".json" file ending to show that a file is in this format. This makes JSON great for both people and programs to work with.

When to use JSON ?

Data exchange format



JSON Syntax

- In JSON , data is primarily stored in two structures: **objects** and **arrays.**
- **1. Using ‘Objects’**
- Objects in JSON are collections of key/value pairs enclosed in curly braces {} .
- Each key is a string (enclosed in double quotes ") followed by a colon : , and the key/value pairs are separated by commas (,) .
- **Example: {"firstName": "John", "lastName": "Doe", "age": 30}**
- **2. Using ‘Arrays’**
- Arrays are ordered lists of values, enclosed in square brackets [] .
- Values within an array are separated by commas (,) .
- **Example: ["apple", "banana", "cherry"]**

Data Format of JSON & XML

JSON

```
{"students":[
  {"name":"Ram", "age":"23", "city":"Agra"},
  {"name":"Sonam", "age":"28", "city":"Delhi"},
]}
```

XML

```
<students>
 <student>
  <name>Ram</name> <age>23</age> <city>Agra</city>
 </student>
 <student>
  <name>Sonam</name> <age>28</age> <city>Delhi</city>
 </student>
</students>
```

Difference Between JSON & XML

JSON

1) JavaScript Object Notation

2) Text Based Format

3) Light Weight

4) Does not support comments and namespaces

XML

1) Extensible Markup Language

2) Markup Language

3) Heavier

4) Supports comments and namespaces

Javscript Object Literals Vs JSON

JavaScript Object

```
var person = { 'first Name' : 'Yahoo', 'last Name' : 'Baba' };

alert(person.firstName + " " + person.lastName);
```

JSON

```
var person = { "firstName" : "Yahoo", "lastName" : "Baba" };

alert(person.firstName + " " + person.lastName);
```

Data types allowed in JSON

- String
- Number
- Boolean
- Array
- object
- NULL

```
{
    "name" : "Yahoo Baba",
    " age" : 25,
    " married " : true,
    " kids " : ,
    " hobbies " : [" music ", " sports "],
    " vehicle " : {
        {" type " : " car ", " vname " : " swift " },
        {" type " : " bike ", " vname" : " pulsar " }
    }
};
```

Advantages of JSON :

- Human Readable Format
- Language Independent
- Supports all Popular Programming languages
- Easy to organised and access
- It is light-weight

Can not use it for transfer video, audio, images or any other binary information.

Features of JSON:

- **Easy to understand:** JSON is easy to read and write.
- **Format:** It is a text-based interchange format. It can store any kind of data in an array of video, audio, and image anything that you required.
- **Support:** It is light-weighted and supported by almost every language and OS. It has a wide range of support for the browsers approx each browser supported by JSON.
- **Dependency:** It is an Independent language that is text-based. It is much faster compared to other text-based structured data.

Advantages of JSON:

1. JSON stores all the data in an array so data transfer makes easier. That’s why JSON is the best for sharing data of any size even audio, video, etc.
2. Its syntax is very easy to use. Its syntax is very small and light-weighted that’s the reason that it executes and response in a faster way.
3. JSON has a wide range for the browser support compatibility with the operating systems, it doesn’t require much effort to make it all browser compatible.
4. On the server-side parsing the most important part that developers want, if the parsing will be fast on the server side then the user can get the fast response, so in this case JSON server-side parsing is the strong point compare tot others.

Disadvantages of JSON:

- The main disadvantage for JSON is that there is **no error handling in JSON,** if there was a slight mistake in the JSON script then you will not get the structured data.
- JSON becomes quite dangerous when you used it with some unauthorized browsers. Like JSON service return a JSON file wrapped in a function call that has to be executed by the browsers if the browsers are unauthorized then your data can be hacked.
- JSON has limited supported tools that we can use during JSON development.

Limitations of JSON

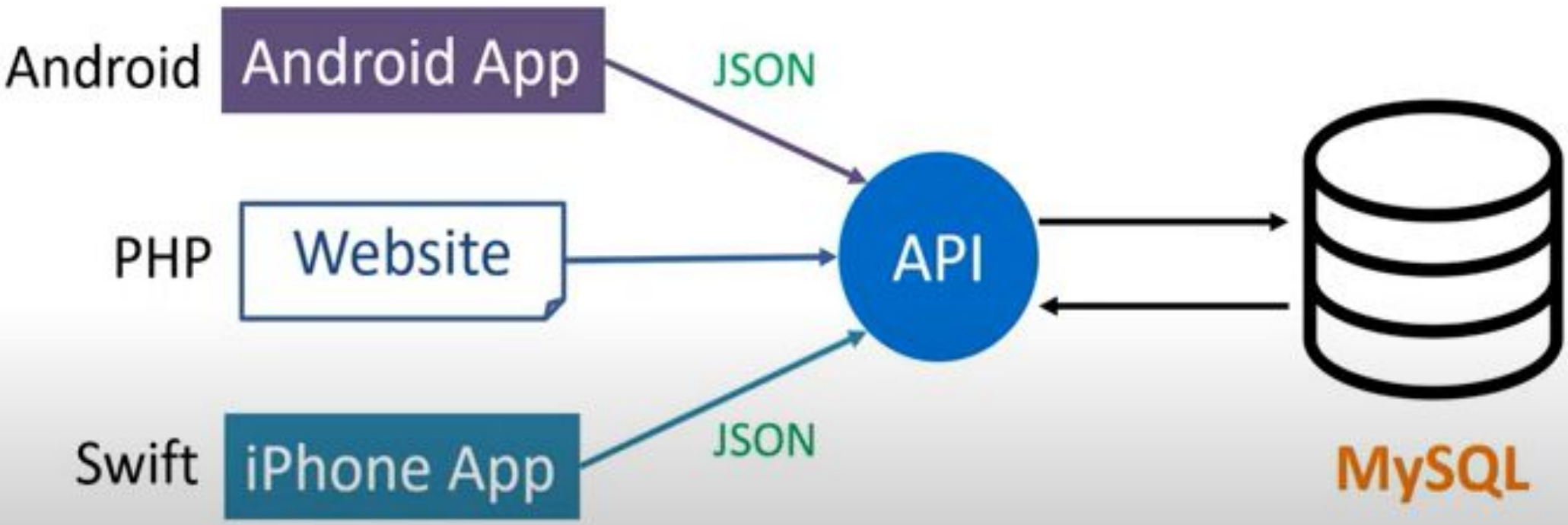
- The main limitation is that there is **no error handling** . If there was a slight mistake in the script then you will not get the structured data.

- It becomes quite **dangerous** when you used it **with some unauthorized browsers** . Like JSON service return a JSON file wrapped in a function call that has to be executed by the browsers if the browsers are unauthorized then your data can be hacked.

- It has **limited supported tools** that we can use during the development.

What is API ?

Application Program Interface



Working with JSON in Applications

- JSON’s true power lies in its ability to seamlessly exchange data between applications. Here are some common use cases:

- **APIs (Application Programming Interfaces):** APIs often use JSON to format data requests and responses. This makes it easier for different applications to communicate and share information.
- **Client-Server Communication:** Web applications frequently use JSON to send data from the browser (client) to the server and vice versa. JSON’s lightweight nature ensures efficient data transfer.
- **Data Storage:** JSON can be used to store configuration settings, user preferences, or any other structured data in a file.