Http Requests & Fetch

Rujuan Xing

# Maharishi International University - Fairfield, Iowa

Wholeness Statement

- In this lecture we introduce the basic technologies that make up the Internet, the World Wide Web and the Hyper Text Markup Language (HTML). We will see that many technologies are built on top of other technologies.

- *Life is found in layers and the TM Technique gives us access to the full range of our awareness and thoughts.*

The Internet

- A connection of computer networks using the Internet Protocol (IP)

- layers of communication protocols: IP → TCP/UDP → HTTP/FTP/POP/SMTP/H...

- What's the difference between the Internet and the World Wide Web (WWW)?

    - The Web is the collection of web sites and pages around the world; the Internet is larger and also includes other services such as email, chat, online games, etc.
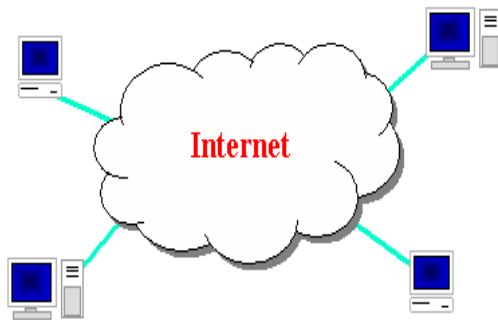
Internet Protocol (IP) IPv4

- The underlying system of communication for all data sent across the Internet.

- Each device has a 32-bit IP address written as four 8-bit numbers (0-255)

- There are two types of IP addresses, servers used to have static IP address while users usually get a dynamic IP address from their ISP.

- Find out your local IP address: in a terminal, type: `ipconfig` (Windows) or `ifconfig` (Mac/Linux)
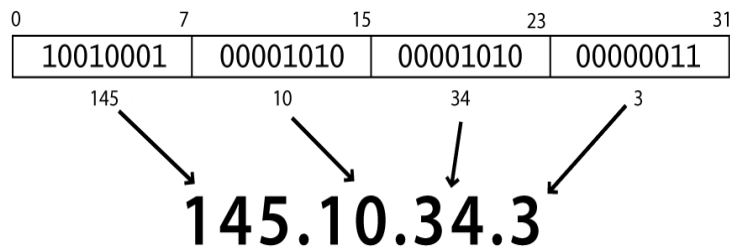
- IPv6 addresses are 128-bit IP address written in hexadecimal and separated by colons. An example IPv6

- address could be written like this: 3ffe:1900:4545:3:200:f8ff:fe21:67cf

Transmission Control Protocol (TCP)

- Adds multiplexing, guaranteed message delivery on top of IP

- **Multiplexing**: multiple programs using the same IP address

  - **port**: a number given to each program or service

  - port 80: web browser (port 443 for secure browsing)

  - port 25: email

  - port 22: ssh

  - Port 21: File transfer (FTP)

  - more common ports

- Some programs (games, streaming media programs) use simpler UDP protocol instead of TCP

Domain Name System (DNS)

- A set of servers that map written names to IP addresses

- Example: www.cs.mum.edu → 69.18.50.54

- Many systems maintain a local DNS cache called a host file:
    - Windows: `C:\Windows\system32\drivers\etc\hosts`
    - Mac: `/private/etc/hosts`
    - Linux: `/etc/hosts`

Hypertext Transport Protocol (HTTP)

- **HTTP** is a <u>protocol</u> which allows the fetching of resources, such as HTML documents. It is the foundation of any data exchange on the Web and it is a client-server protocol, which means requests are initiated by the recipient, usually the Web browser.
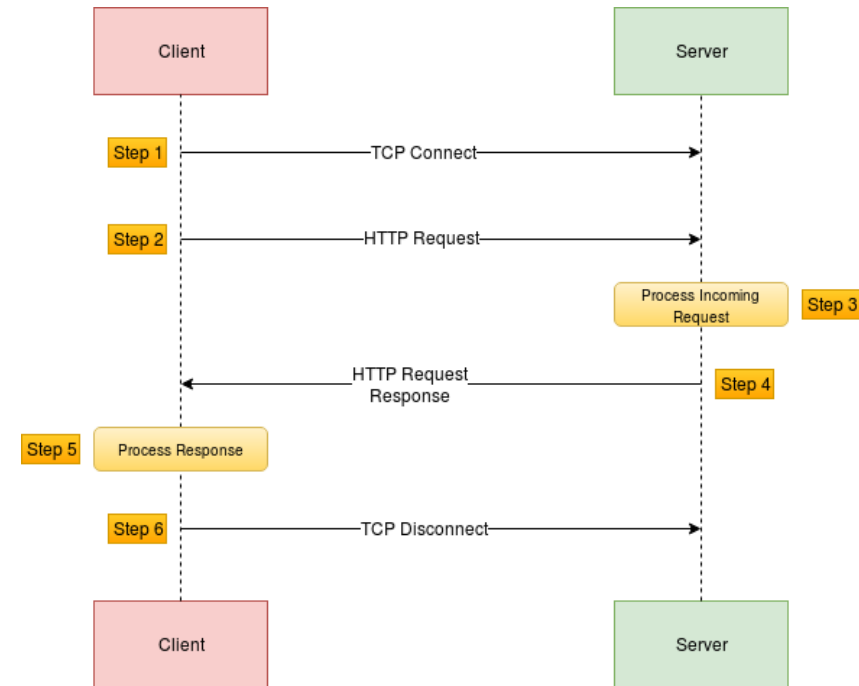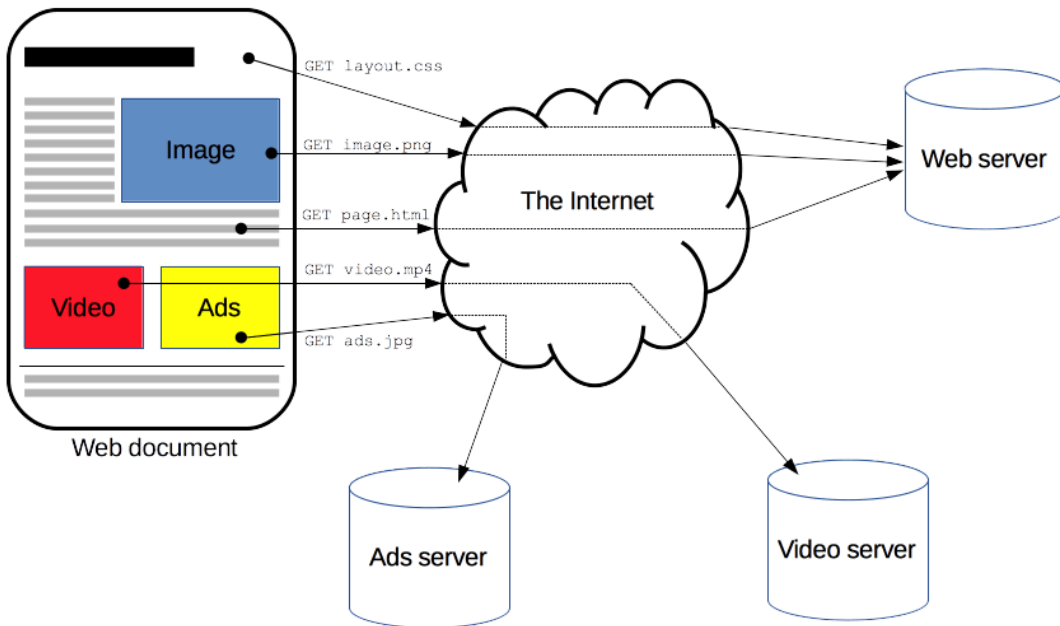
How HTTP Protocol Works

How browsers display a webpage

1. User machines have IP address on the Internet

2. Server machines have IP address and Domain Name

3. Domain names and IP addresses are registered at global DNS Server

4. When the user opens a browser window and asks for www.test.com

5. First, the browser will check the local DNS (host file) for the IP address of that domain

6. If not found, it will connect to ISP and ask it for the DNS

7. Once retrieved, the browser will send another request to that server

8. Requests are delivered by the IP protocol, collected by the TCP protocol, and processed by HTTP or HTTPS protocol

9. The server will send the browser a response with HTML code.

10. The browser will interpret the HTML code line by line and start building the web page.

11. For every resource not found in the browser cache, the browser will send a new request to the server again asking for that resource and so on.

HTTP Request & Response Example

Requests

```
POST / HTTP/1.1
Host: localhost:8000
User-Agent: Mozilla/5.0 (Macintosh;… )… Firefox/51.0
Accept: text/html,application/xhtml+xml,…,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data; boundary=-12656974
Content-Length: 345

-12656974
(more data)
```

Responses

```
HTTP/1.1 403 Forbidden
Server: Apache
Content-Type: text/html; charset=iso-8859-1
Date: Wed, 10 Aug 2016 09:23:25 GMT
Keep-Alive: timeout=5, max=1000
Connection: Keep-Alive
Age: 3464
Date: Wed, 10 Aug 2016 09:46:25 GMT
X-Cache-Info: caching
Content-Length: 220

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML
2.0//EN">
(more data)
```

start-line

HTTP headers

empty line

body

HTTP Methods/Commands
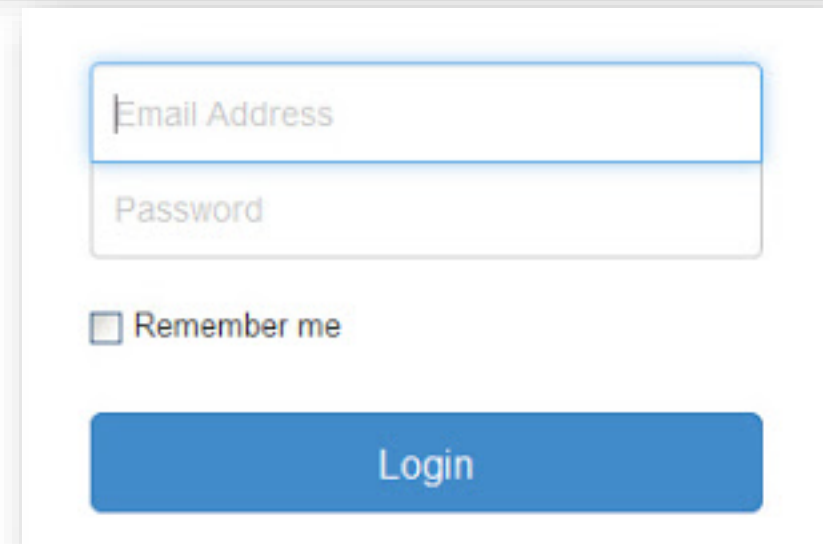
- **GET**: Requests a specific file or resource from the server

- **POST**: Submits form information to the server

- **PUT**: Uploads a file to the server

- **DELETE**: Delete data from the server

- **HEAD**: Requests information about a file from the server, but not the file's entire contents.

- **PATCH**: Partially update a certain data on the server

- **OPTIONS**: Handshaking and retrieves the capabilities of the server

How to send a Request?

- URL (**GET** only) ⟶
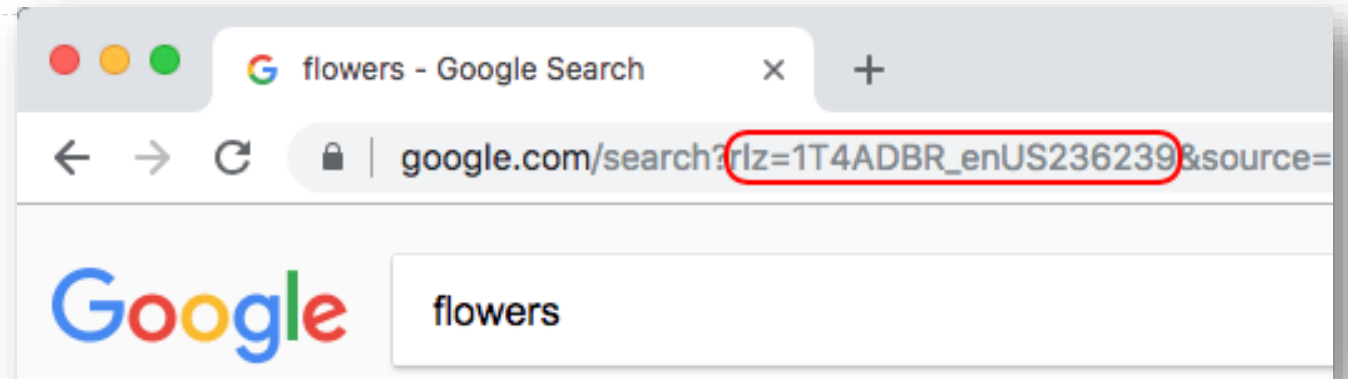
- Form element (**GET** and **POST** only) ⟶

How to send a Request?

- JavaScript (AJAX) with Fetch API (supports all verbs)

```
fetch('https://dog.ceo/api/breeds/image/random')
.then(response => {
  console.log(response.headers.get('Content-Type'))
  console.log(response.headers.get('Date'))
})
```

▶ *Promise {<pending>}*

application/json

Sun, 22 Apr 2018 05:05:00 GMT

Uniform Resource Locator (URL)

- URL stands for Uniform Resource Locator. A URL is nothing more than the address of a given unique resource on the Web.

href									

protocol		auth			host		path		hash
					hostname	port	pathname	search	
								query	
" https:	//	user	:	pass	@ sub.host.com hostname	: 8080 port	/p/a/t/h	? query=string	#hash "
protocol		username	password		host				
origin					origin		pathname	search	hash

href									

- **Anchor**: jumps to a given section of a web page - #hash

- **Query String**: a set of parameters passed to a web program - ?query=string

HTTP Limitations

- Security Concerns

    - Privacy: anyone can see content

    - Integrity: Someone might alter content. HTTP is insecure since no encryption methods are used. Hence is subject to man in the middle and eavesdropping of sensitive information

    - Authentication: Not clear who you're talking with. Authentication is sent in the clear – Anyone who intercepts the request can determine the username and password being used.

- Stateless – need statement management techniques to maintain the information across multiple request-response cycles.

Hypertext Transfer Protocol Secure HTTPS

- HTTPS (Hypertext Transfer Protocol Secure) is a secure version of the HTTP protocol that uses the Secure Sockets Layer(SSL)/Transport Layer Security(TLS) protocol for encryption and authentication.

- HTTPS by default uses port 443 as opposed to the standard HTTP port of 80
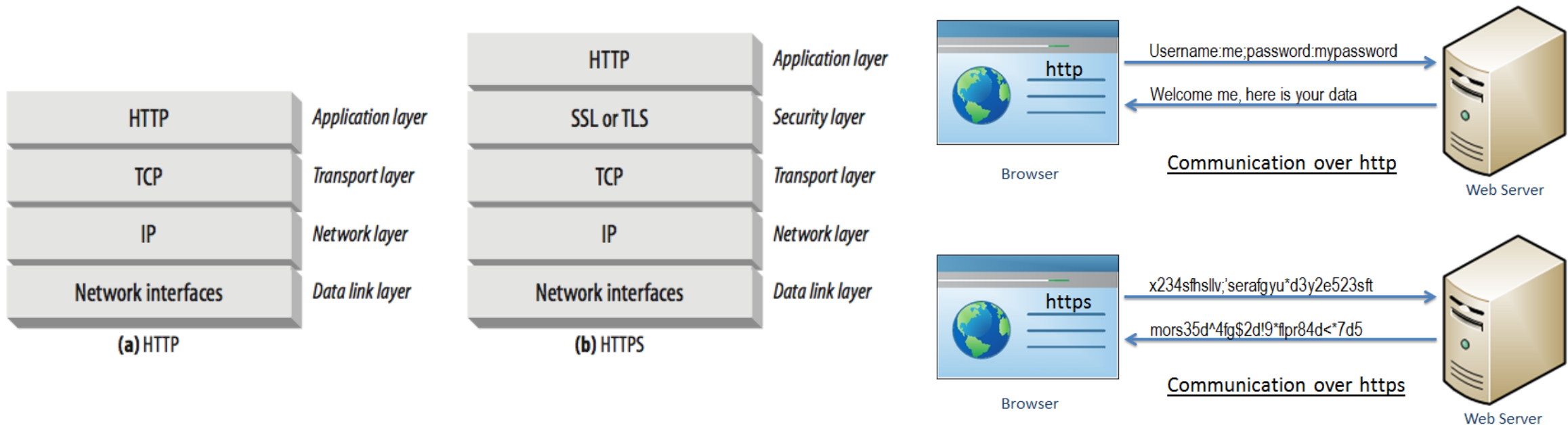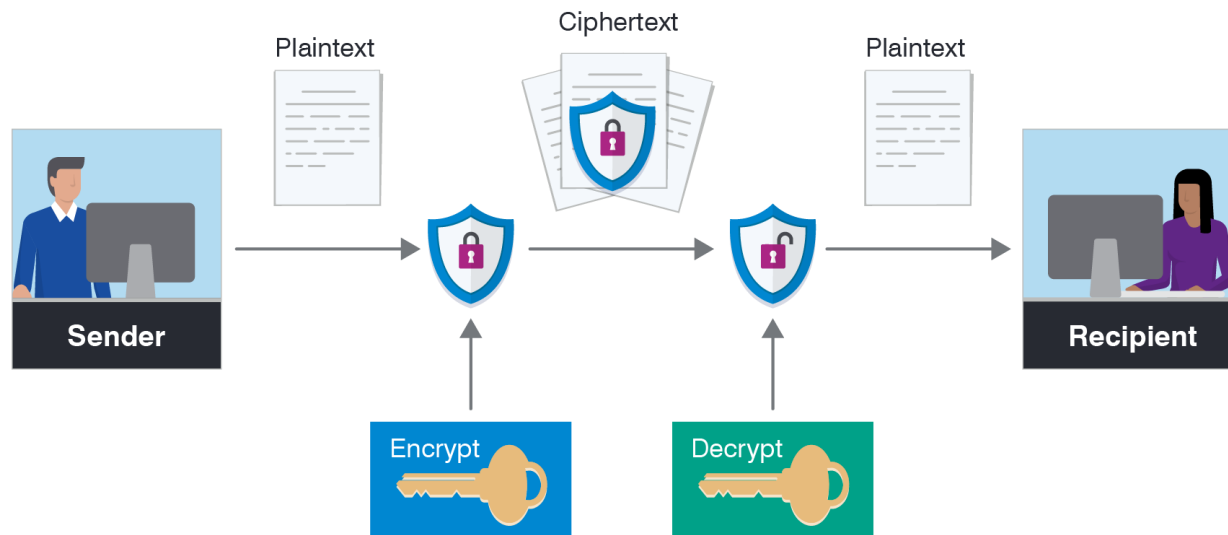
TLS

- Transport Layer Security, or TLS, is a widely adopted security protocol designed to facilitate privacy and data security for communications over the Internet.

- A primary use case of TLS is encrypting the communication between web applications and servers, such as web browsers loading a website.

- TLS can also be used to encrypt other communications such as email, messaging, and voice over IP (VoIP).

- 3 main components which TLS accomplished:

  - **Encryption**: hides the data being transferred from third parties.

  - **Authentication**: ensures that the parties exchanging information are who they claim to be.

  - **Integrity**: verifies that the data has not been forged or tampered with.
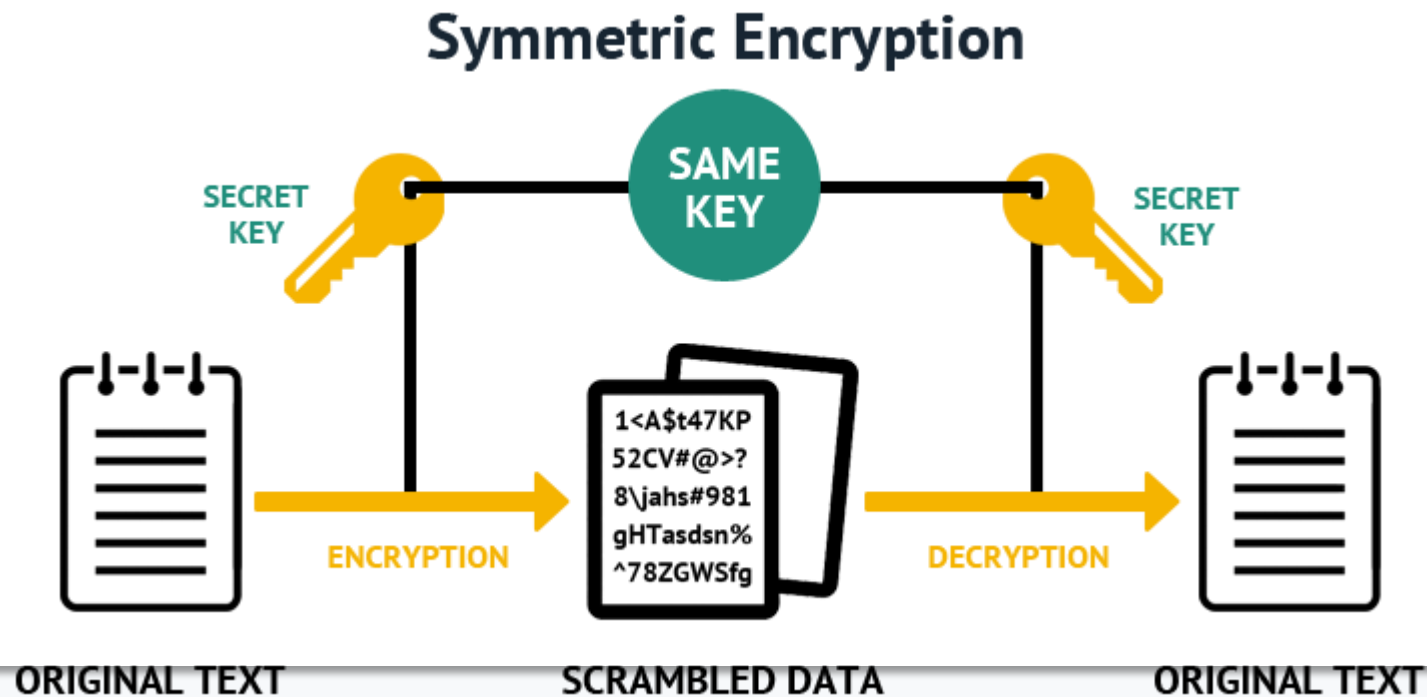
Encryption

- In cryptography, encryption is the process of encoding a message or information in a way that only authorized parties can access it and those who are not authorized cannot.

- Example: base64, RSA, Crypto, AES

"Hello" → encryption → "SNifgNi+uk0="

plaintext   ciphertext

Plaintext   Ciphertext   Plaintext

Sender   Recipient

Encrypt   Decrypt

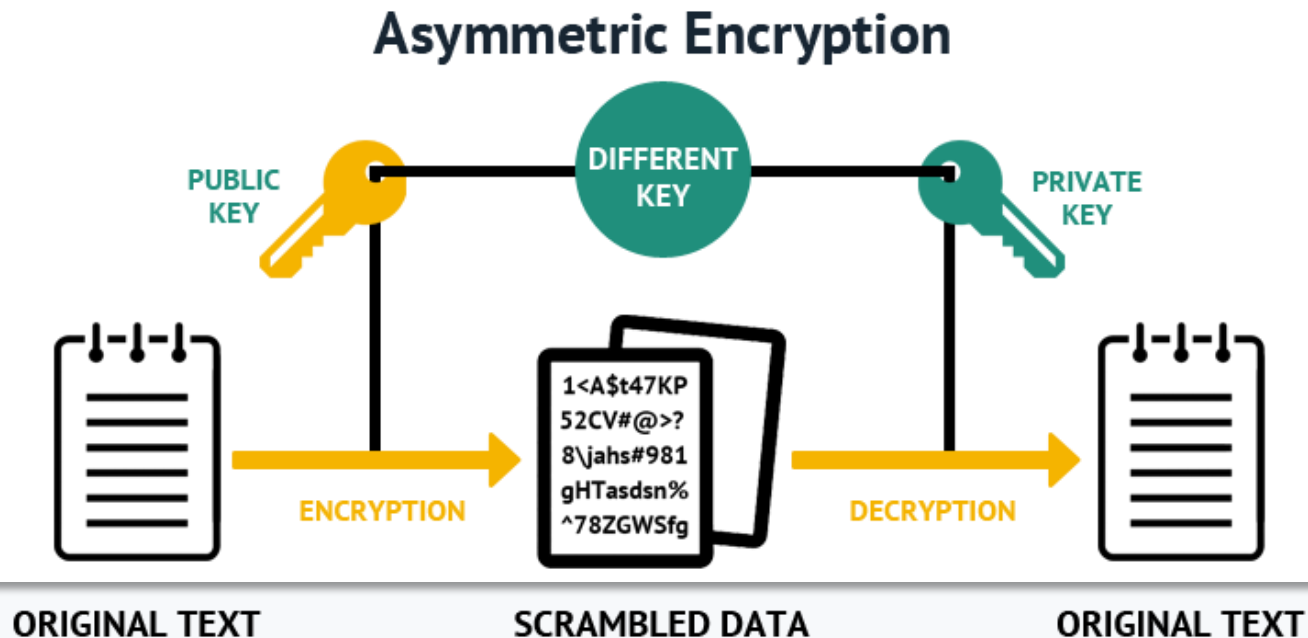Different keys are used to encrypt and decrypt messages

Encryption Types / Methods - Symmetric Encryption

- Symmetric Encryption also called `Secret Key Cryptography`, it employs the same secret key for both encryption and decryption, that is used to exchange information during a secure session between the client's browser and web server with an SSL Certificate.

- Symmetric encryption uses 128 or 256 bits key, based on the security requirement.
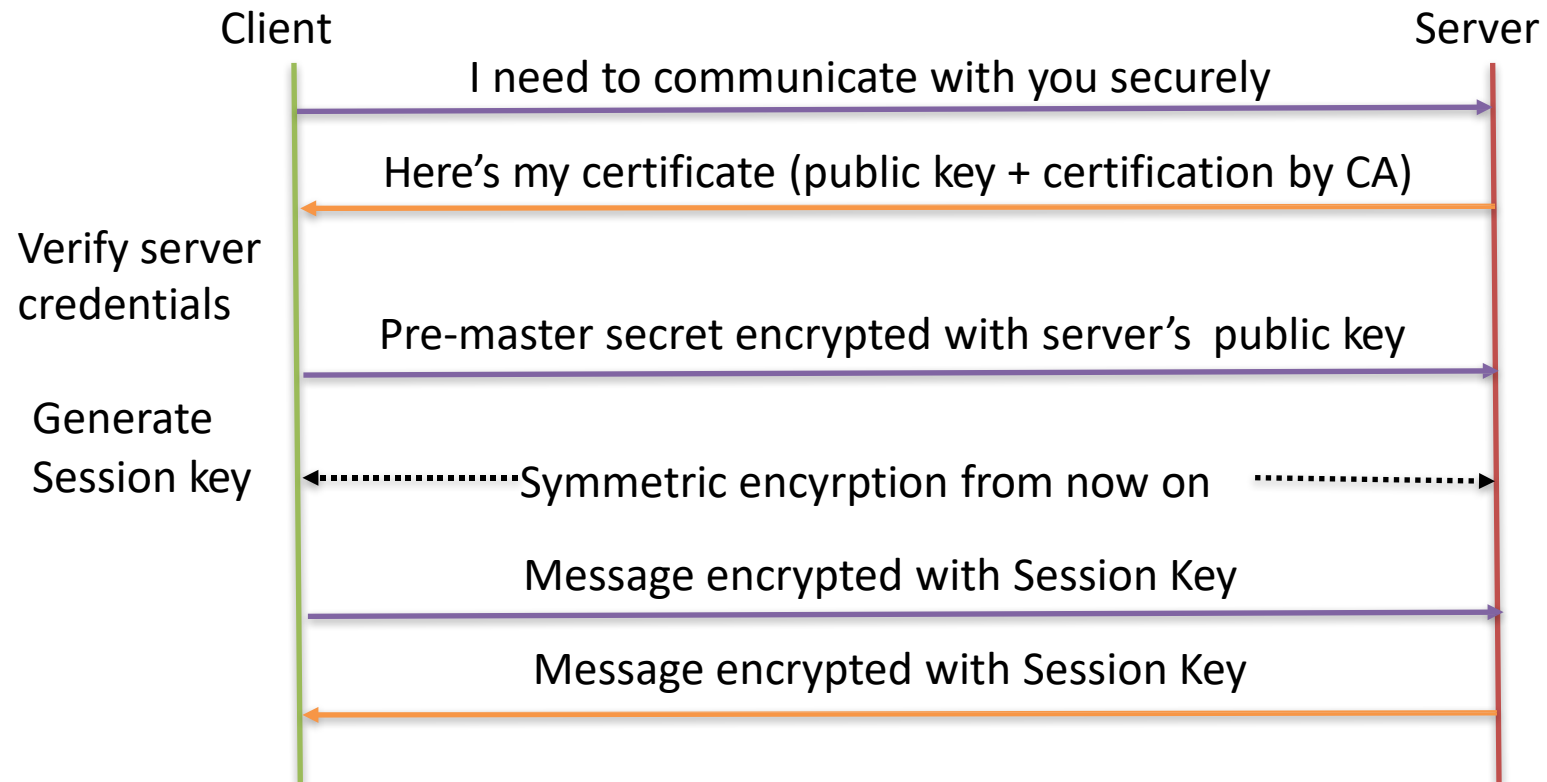
Encryption Types / Methods - Asymmetric Encryption

- Asymmetric Encryption also called as Public Key Cryptography and it uses two different keys – a public key used for encryption and a private key used for decryption, that is used in SSL handshake process.

- The public key can be made freely available to any person who might be interested in sending a message, the private key remains a secret well kept by the receiver of the message.

- Asymmetric encryption uses much larger like 2048 bits RSA Keys, based on the security requirement.

Asymmetric Encryption

PUBLIC KEY — DIFFERENT KEY — PRIVATE KEY

ORIGINAL TEXT → ENCRYPTION → SCRAMBLED DATA (1<A$t47KP 52CV#@>? 8\jahs#981 gHTasdsn% ^78ZGWSfg) → DECRYPTION → ORIGINAL TEXT

TLS Handshake

Digital Certificate

A certificate is an electronic credentials used to assert the online identities of individuals, applications and other entities on a network.

They are similar to an ID card (passport, driving license, diploma). A certificate is signed by (Certificate Authority CA).

- A certificate is a file contains:

- Identity of owner

- Public Key

- Signature of the certificate issuer

- Expiration date

A signature is always made with the private key of issuer and can be validated with the public key of the issuer.

Anyone with access to the server's public key, can verify that a signature could only have been created by the server's private key.
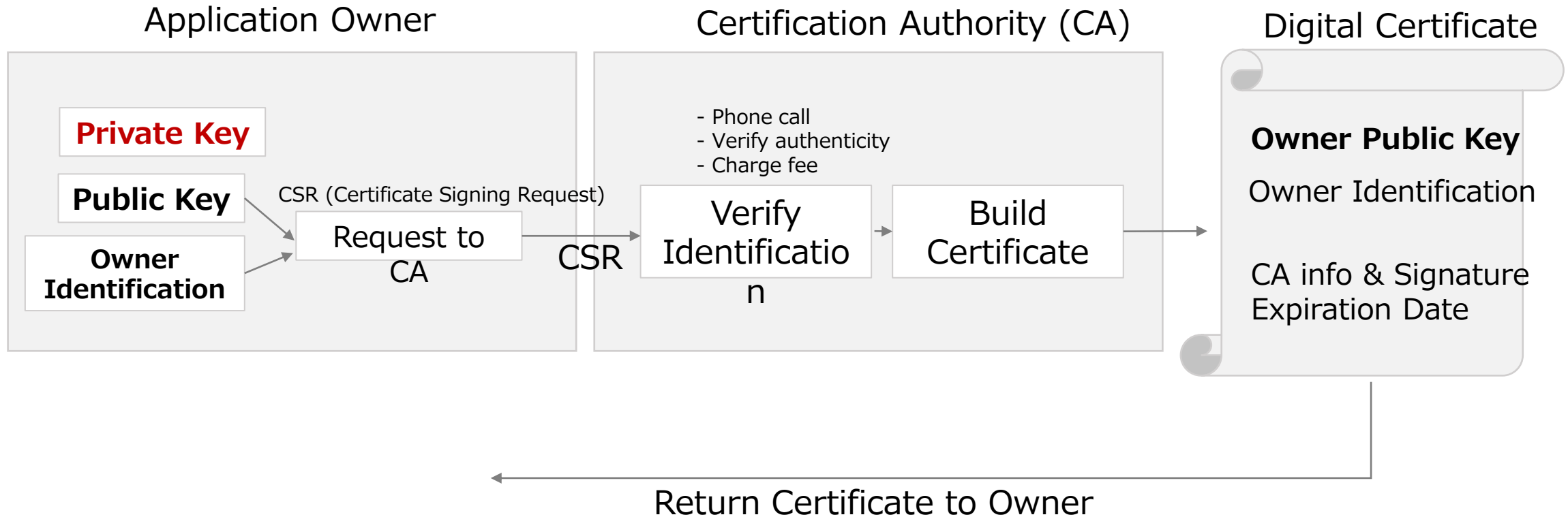
Certificate Issuer

A certificate authority (CA) validates the identity of the certificate holder.

There are two types of CA:

- **RCA**: Trusted Root Certification Authorities (CA): Every device includes something called a root store. A root store is a collection of pre-downloaded root certificates (and their public keys) that live on the device itself.

  - Apple users, both macOS and iOS, rely on the Apple root store, likewise for Microsoft users and its root store. Android uses Google's. And the Mozilla suite of products uses its own proprietary root store.

- **ICA**: Intermediate Certification Authorities (They own certificates from CA) (a security layer so private key for RCA is inaccessible)

How Digital Certificates Are Created?

Application Owner

Private Key

Public Key

Owner Identification

CSR (Certificate Signing Request)

Request to CA

CSR

Certification Authority (CA)

- Phone call
- Verify authenticity
- Charge fee

Verify Identification

Build Certificate

Digital Certificate

Owner Public Key

Owner Identification

CA info & Signature
Expiration Date

Return Certificate to Owner

Using Digital Certificates

When browsers contact an application owner via TLS, the first thing they receive is the digital certificate. Browsers check the validity of this certificate (if it was issued to the claimed app owner, not expired, the CA signature).

Usually a **chain of certificates** are sent to the browser (the owner certificate signed by ICA, and ICA certificate signed by Root CA), since the browser has the public key for all Root CAs, it validates the ICA certificate first and make sure it is a certified entity, and then use it's public key in that certificate to validate the owner certificate.
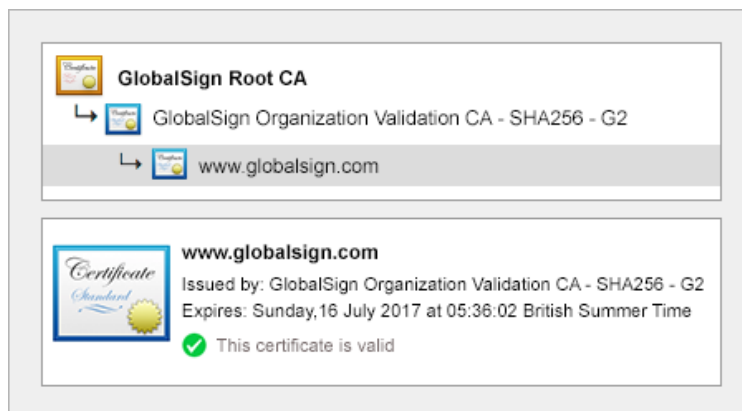
Main Point

The internet is a global computer network that uses IP addresses to uniquely identify computers on the network. Through the TCP protocol each IP address can work with multiple services at the same time. One of these services is the HTTP protocol which is used by the World Wide Web to transport HTML pages.

These are many layers of the internet. Familiarity with the deep layers is necessary for optimal understanding and use of the surface layers. TM settles the mind so that it is more connected to the deeper layers that are the basis of the more expressed layers.

Web Application and AJAX

- **Web Application**: a dynamic web site that mimics the look and feel of a desktop application. Presents a continuous user experience rather than disjoint pages

- **AJAX**: Asynchronous JavaScript and XML

  - Not a programming language, but another asynchronous JavaScript API

  - Downloads data from a server in the background

  - Allows dynamically updating a page without making the user wait

  - Avoids the "click-wait-refresh" pattern

JavaScript Object Notation (JSON)

- JSON (JavaScript Object Notation) is a lightweight data-interchange format.

  - Based on a subset of the JavaScript Programming Language Standard ECMA-262 3rd Edition - December 1999.

  - A text format that is completely language independent.

  - Easy for machines to parse and generate.

    - Can convert any JavaScript object into JSON, and send JSON to the server.

  - Natively supported by all modern browsers

  - Replaced XML (Extensible Markup Language)

XML vs JSON

- Both JSON and XML can be used to share data between client and server.

- The biggest difference is:

- XML has to be parsed with an XML parser. JSON can be parsed by a standard JavaScript function.

- XML is much more difficult to parse than JSON.

```xml
<empinfo>
    <employees>
        <employee>
            <name>Scott Philip</name>
            <salary>£44k</salary>
            <age>27</age>
        </employee>
        <employee>
            <name>Tim Henn</name>
            <salary>£40k</salary>
            <age>27</age>
        </employee>
        <employee>
            <name>Long yong</name>
            <salary>£40k</salary>
            <age>28</age>
        </employee>
    </employees>
</empinfo>
```

```json
{ "empinfo" :
    {
        "employees" : [
        {
            "name" : "Scott Philip",
            "salary" : £44k,
            "age" : 27,
        },
        {
            "name" : "Tim Henn",
            "salary" : £40k,
            "age" : 27,
        },
        {
            "name" : "Long Yong",
            "salary" : £40k,
            "age" : 28,
        }
        ]
    }
}
```

JavaScript Object Notation (JSON)

- JSON is a syntax similar to JS Objects for storing and exchanging data and an efficient alternative to XML.
- A name/value pair consists of a field name **in double quotes**, followed by a colon, followed by a value. Values can be any JS valid type except functions.

```
{"students":[
    {"firstName":"Ashim", "lastName":"Ghimire"},
    {"firstName":"Mohamed", "lastName":"Hassan"},
    {"firstName":"Leul", "lastName":"Necha"},
    {"firstName":"Shawn", "lastName":"Daudi"}
]}
```

- JSON values can be:
  - A number (integer or floating point)
  - A string (in double quotes)
  - A Boolean (true or false)
  - An array (in square brackets)
  - An object (in curly braces)
  - null

Browser JSON Methods

Method	Description
JSON.parse(*string***)**	Converts the given string of JSON data into an equivalent JavaScript object and returns it
JSON.stringify(*object***)**	Converts the given object into a string of JSON data (the opposite ofJSON.parse)

JSON expressions exercise

```
const jsonString = '{
  "window": {
    "title": "Sample Widget",
    "width": 500,
    "height": 500
  },
  "image": {
    "src": "images/logo.png",
    "coords": [250, 150, 350, 400],
    "alignment": "center"
  },
  "messages": [
    {"text": "Save", "offset": [10, 30]}
    {"text": "Help", "offset": [ 0, 50]},
    {"text": "Quit", "offset": [30, 10]},
  ],
  "debug": "true"
}';
const data = JSON.parse(jsonString);
```

Given the JSON data at right, what expressions would produce:

Using JavaScript Syntax on `data` object.

- The window's title?

```
let title = data.window.title;
```

- The image's third coordinate?

```
let coord = data.image.coords[2];
```

- The number of messages?

```
let len = data.messages.length;
```
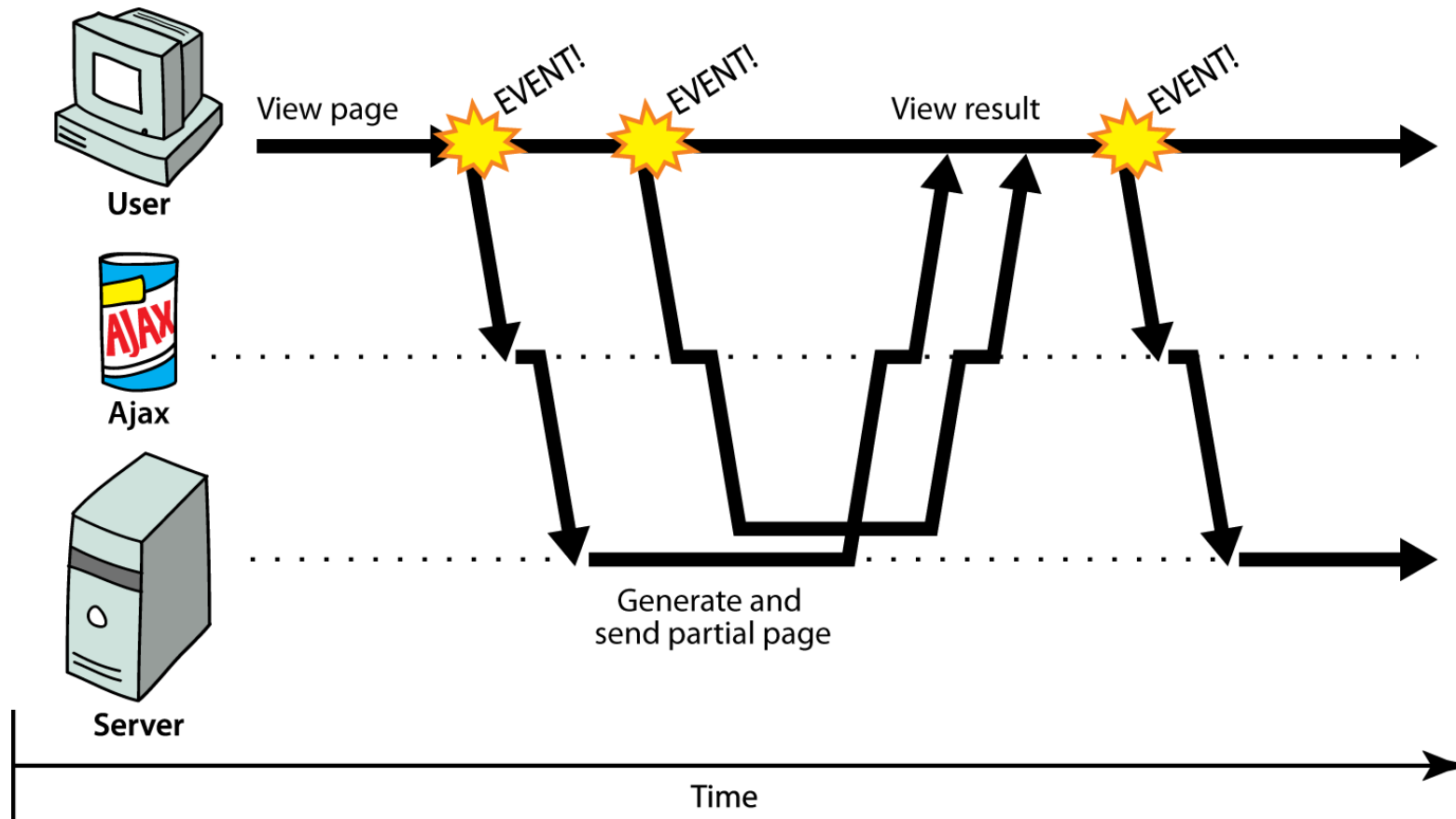
- The y-offset of the last message?

```
let y = data.messages[len
1].offset[1];
```
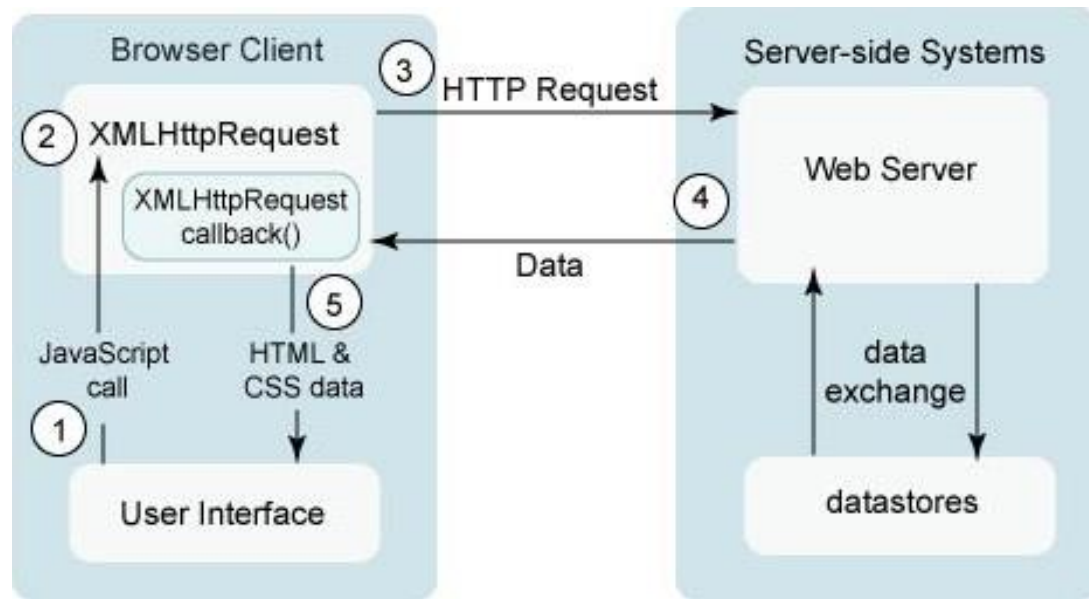
Asynchronous Web Communication

Users keep interacting with page while data loads

AJAX Calls

- The browser will send an asynchronous request to the server and register a callback function to handle the response.

- When servers send the response, it will be passed to the callback function and pushed into the Macrotask queue.

Fetch API

- The Fetch API provides a JavaScript interface for accessing and manipulating parts of the HTTP pipeline, such as requests and responses. It also provides a **global** fetch() method that provides an easy, logical way to fetch resources **asynchronously** across the network.

- This kind of functionality was previously achieved using `XMLHttpRequest`.

- Syntax:

```
let promise = fetch(url, [options]);
```

  - `url` – the URL to access.

  - `options` – optional parameters: method, headers etc.

  - Without `options`, this is a simple `GET` request, downloading the contents of the url.

Using fetch() - GET

```
let postPromise = fetch('https://jsonplaceholder.typicode.com/posts/101')
```

The browser starts the request right away and returns a promise that the calling code should use to get the result.

- Getting a response is usually a two-stage process.
  - First, the promise, returned by fetch, resolves with an object of the built-in Response class as soon as the server responds with headers.
    - `postPromise.then(response => console.log(response.ok, response.status));`

  - Second, to get the response body, we need to use an additional method call.
    - response.text() – read the response and return as text,
    - response.json() – parse the response as JSON,
    - …
    ```
    postPromise.then(response => response.json())
            .then(result => console.log(result));
    ```

Using fetch() – POST with options

```javascript
fetch('https://jsonplaceholder.typicode.com/posts', {
      method: 'POST',
      body: JSON.stringify({
          title: 'cs445',
          body: 'This is CS445 Lesson 7 Demo',
          userId: 1,
      }),
      headers: {
          'Content-type': 'application/json; charset=UTF-8',
      },
})
.then((response) => response.json())
.then((json) => console.log(json));
```
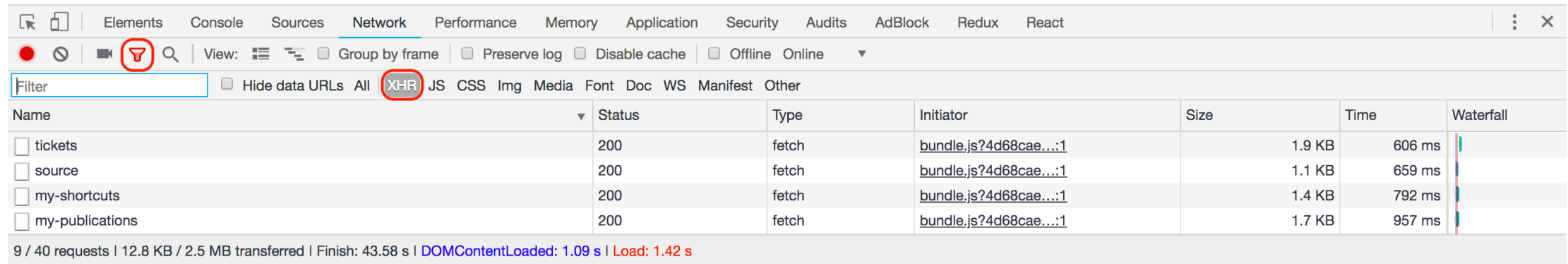
Using Async & Await

```javascript
async function fetchPostById(postid) {
    let response = await fetch("https://jsonplaceholder.typicode.com/posts/" + postid);
    if (response.ok) {
        let json = await response.json();
        console.log(json);
    } else {
        console.log("HTTP-Error: " + response.status);
    }
}


fetchPostById(1);


// Remember to use try/catch in case of network problem
```

Debugging AJAX Requests

Single Page Applications (SPA)

- Your browser sends a request to the server and receives a response with a single HTML page and JavaScript client-side application.

- HTML file has a single outlet element, where All DOM creation, and upcoming requests are handled by JavaScript in the browser (including Routes). The browser sends AJAX calls behind the scene to get/post data from the server when needed.

Postman - The Collaboration Platform for API Development

- Download: https://www.postman.com/downloads/

- https://www.postman.com/

Main Point JSON

JSON has become more widely used for Ajax data representations than XML because JSON is easier to write and read and is almost identical to JavaScript object literal syntax.

Science of Consciousness: We always prefer to do less and accomplish more. Actions arising from deep levels of consciousness are more efficient and effective.

CONNECTING THE PARTS OF KNOWLEDGE WITH THE WHOLENESS OF KNOWLEDGE

Frictionless Flow of Information

1. Client-side programming with JavaScript is useful for making web applications highly responsive.
2. Ajax allows JavaScript to access the server in a very efficient manner using asynchronous messaging and partial page refreshing.

_____

3. **Transcendental consciousness** is the experience of the home of all the laws of nature where all information is available at every point.
4. **Impulses within the transcendental field**: Communication at this level is instantaneous and effortless.
5. **Wholeness moving within itself**: In unity consciousness daily life is experienced in terms of this frictionless and effortless flow of information.

References

- https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages

- https://www.cheapsslshop.com/blog/symmetric-vs-asymmetric-encryption-whats-the-difference/

- https://www.jscape.com/blog/cipher-suites

- https://javascript.info/fetch