Internet and Web Technologies

Computer Science for Lawyers and Humanitarian Workers (ICRC)

Prof. Kévin Huguenin

ADMINISTRATIVE INFORMATION

Tomorrow morning, Prof. Falsafi's lecture on "Cloud and Service Computing" starts at **8:45am** (instead of 9am).

A FEW WORDS ABOUT ME

Kévin Huguenin

Professor (UNIL-HEC Lausanne) – Since 2016

Researcher (CNRS, France)
Post-doctoral Researcher (McGill, Canada; EPFL, Suisse)

PhD Candidate (Inria, France)

Expertise: Networked and Distributed systems, Information Security and Online Privacy
Background: Mathematics and Computer science

LEARNING OUTCOMES AND CONTENTS

- **Learning outcomes:**
  - Understand the general structure and organization of the Internet and of the Web
  - Understand the different technologies (protocols, programming languages and software) underlying the Internet and the Web
    - HTTP, URL, *etc*.
    - HTML, CSS, XML, JSON
    - Browser, web-server

- **Teaching philosophy:**
  - Keep it simple
  - Learn by doing/trying; many do-it-yourself examples (hands-on)

- **Contents**:
  - Web protocols (HTTP)
  - Web content
    - Content, format, modification, and rendering (HTML, CSS and JavaScript)
    - Production (PHP, Python, Ruby)
  - Web sites
    - Content management system (CMS)
  - Web APIs and Web services
  - Search engines
  - The invisible and the dark web

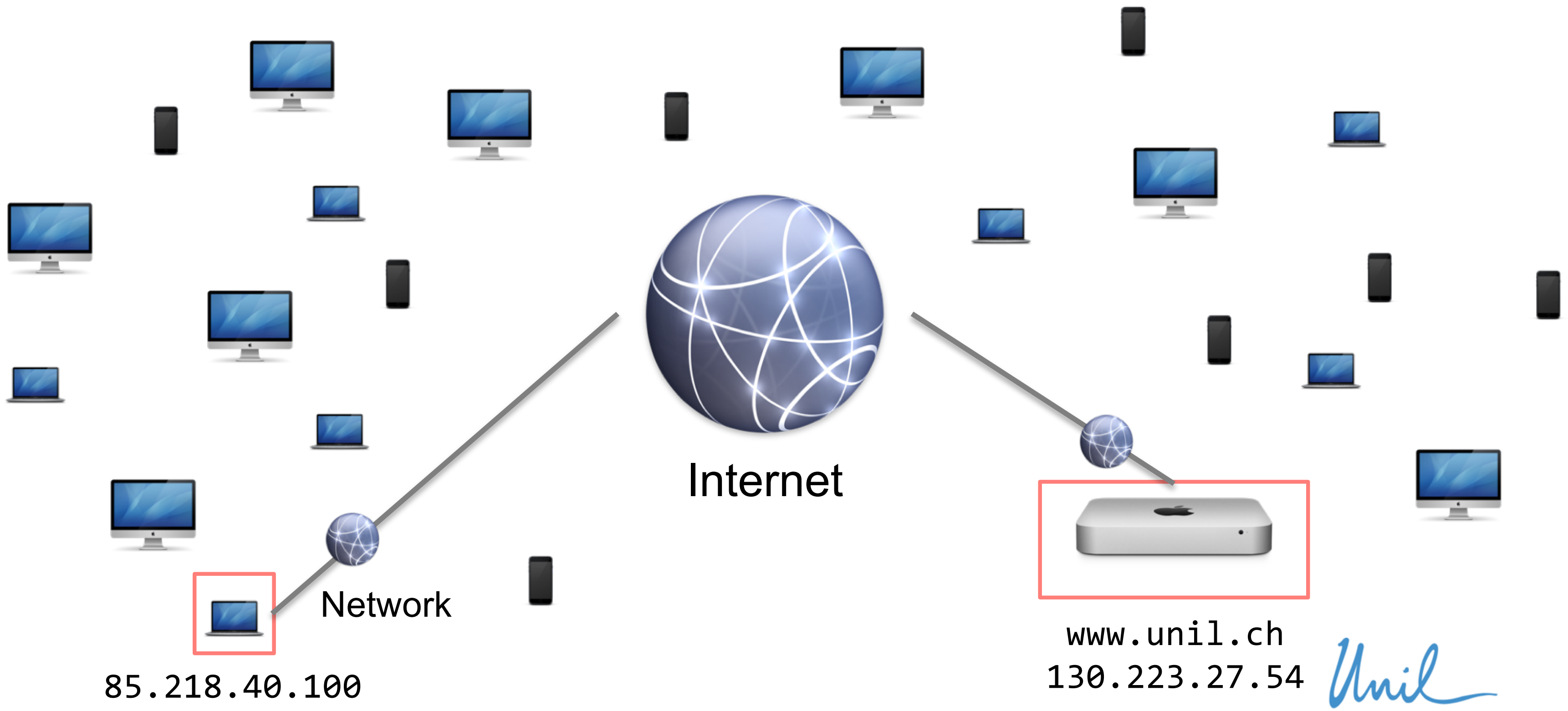
RESOURCES

- The course website on Moodle: https://moodle.epfl.ch/course/view.php?id=15667

- The W3C (World Wide Web consortium) learning website https://www.w3schools.com
- The MDN (Mozilla Developer Network) learning website https://developer.mozilla.org/bm/docs/Web

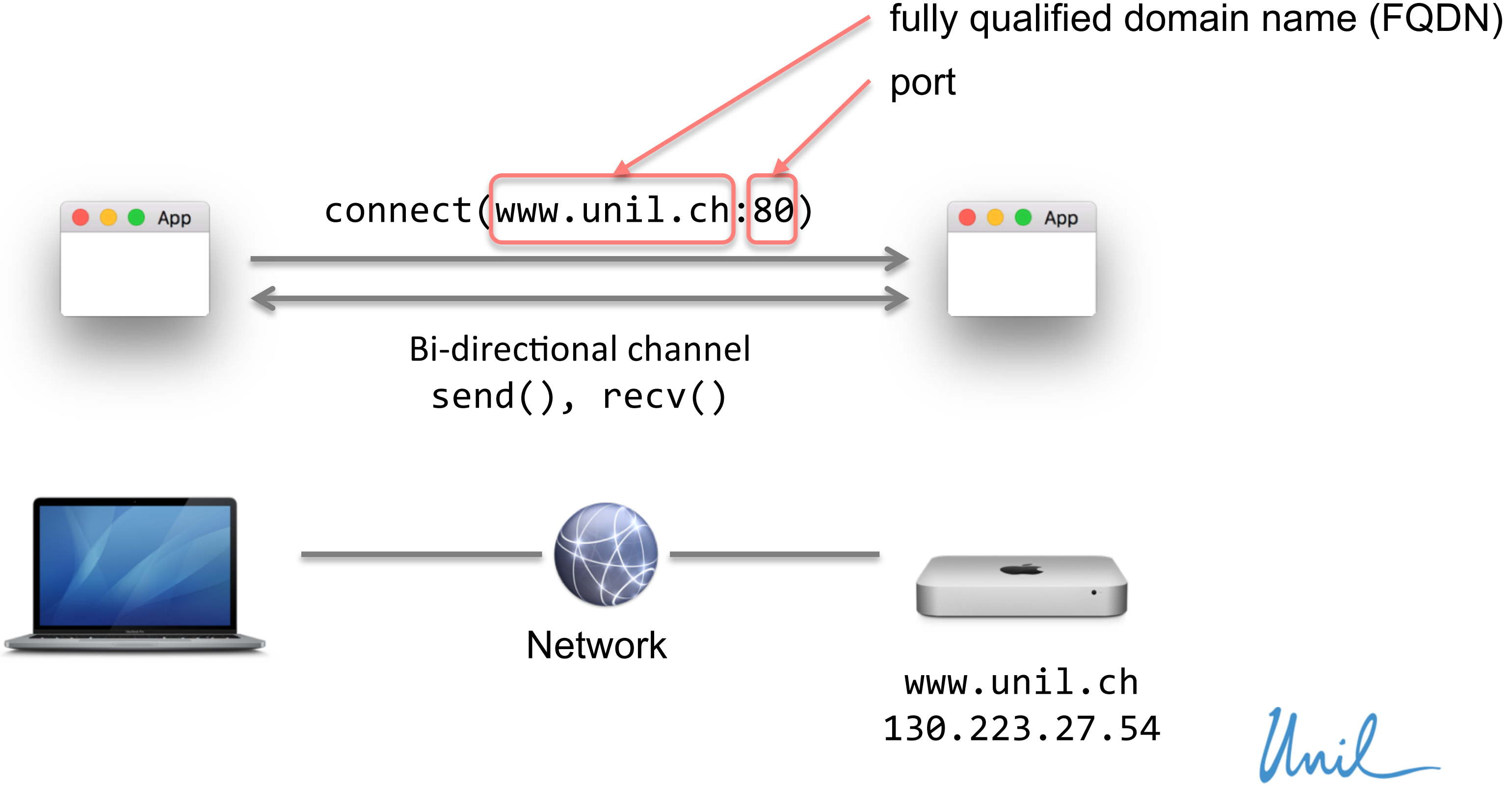
THE INTERNET

- System model: Network of interconnected devices
  - Created in the 60's (ARPANET, DoD project); ~11 billion devices (2018)
  - (Applications running on) devices that can communicate with each other



THE INTERNET

- (TCP/IP) Client-server architecture
- Client connects to the server; it initiates the connection



PROTOCOLS

- A protocol describes how (remote) applications communicate and interact with each other
  - Focus on application-layer communications (top of the network stack)

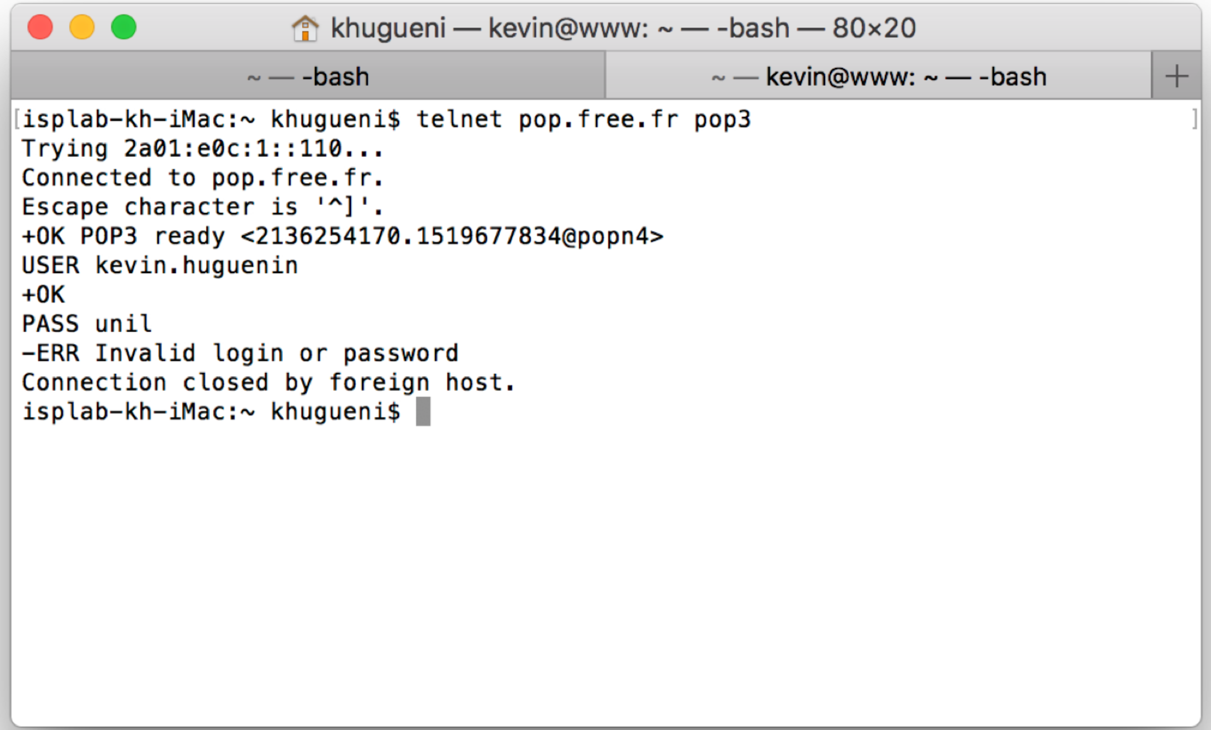
- Internet protocols
  - pop and imap (receiving e-mail), smtp (sending e-mail), rsh/ssh (command line a.k.a. terminal a.k.a. shell), ftp (file transfer), http (web)
  - Standard protocols are usually associated with fixed port (e.g., http: 80)

- Example:
  - [server] Hello, I am a mail server running version X. Please authenticate.
  - [client] Hello, I am user “kevin.huguenin”
  - [server] Please input your password
  - [client] My password is “unil”
  - [server] Your username and password match, welcome!
  - [client] Do I have any new e-mail?

Demo

PROTOCOLS

- Demo with `telnet`
  - `telnet` enables a client to communicate directly with a remote server application (only used for testing!)



```
isplab-kh-iMac:~ khugueni$ telnet pop.free.fr pop3
Trying 2a01:e0c:1::110...
Connected to pop.free.fr.
Escape character is '^]'.
+OK POP3 ready <2136254170.1519677834@popn4>
USER kevin.huguenin
+OK
PASS unil
-ERR Invalid login or password
Connection closed by foreign host.
isplab-kh-iMac:~ khugueni$
```

Démo

THE HTTP PROTOCOL

- The HTTP (*HyperText Transfer Protocol*) protocols specifies how to interact with a web server
  - The web client (application)—typically a web browser (e.g., Safari, Chrome)—initiates the connections and sends a request for a given ressource (GET)
  - The web server (e.g., httpd, apache) executes the requests: It sends back the requested resources

```
$> telnet www.imdb.com http
> GET /
```

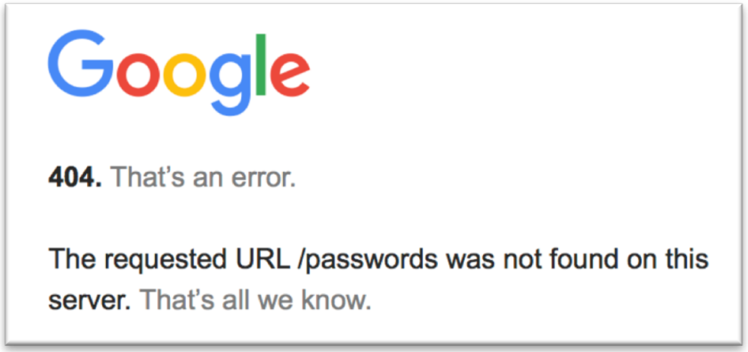
root of the website

```
HTTP/1.1 200 OK
Date: Mon, 26 Feb 2018 21:21:21 GMT
Content-Type: text/html;charset=UTF-8
Content-Language: en-US
```

header

```
<!DOCTYPE html>
<html xmlns:og="http://ogp.me/ns#">
    <head>
        <meta charset="utf-8">
…
```

content



404. That's an error.

The requested URL /passwords was not found on this server. That's all we know.

THE HTTP PROTOCOL

- A few words on HTTP cookies (nothing to do with this:)
  - A web server can ask the web client to create a cookie—typically to keep track of the session of an authenticated user (`set-cookie` in the header)

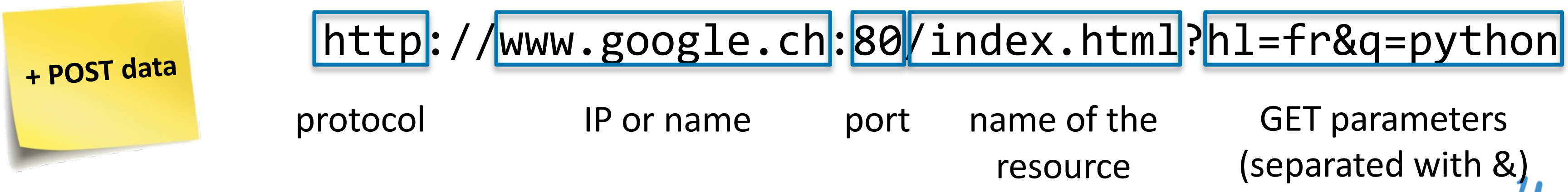
```
[client]
GET /login HTTP/1.0
…

[server]
HTTP/1.0 200 OK
Set-Cookie: sessionToken=xyz
…

[client]
GET /mailbox HTTP/1.0
Set-Cookie: sessionToken=xyz
```

  - Risk of tracking, especially when cookies created for a specific web server are sent to another web server (third-party cookies), e.g., social networks and advertisers.

Démo

THE HTTP PROTOCOL

- Originally, webpages were mostly static (stored in a file, sent untouched upon request)
- Today, webpages are mostly dynamic: they are generated upon request by a program (back-end), based on the parameters specified by the client, e.g., the language
  - The back-end can be programmed in any language (e.g., PHP, Python, Ruby, JavaScript); it usually interacts with databases
- A resource is specified with a URL (*Unified Ressource Location*) of the following form:



+ POST data

```
http://www.google.ch:80/index.html?hl=fr&q=python
```

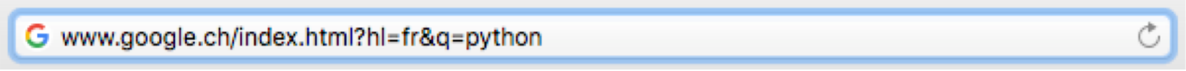
protocol

IP or name

port

name of the resource

GET parameters (separated with &)



www.google.ch/index.html?hl=fr&q=python

Démo

THE HTTP PROTOCOL

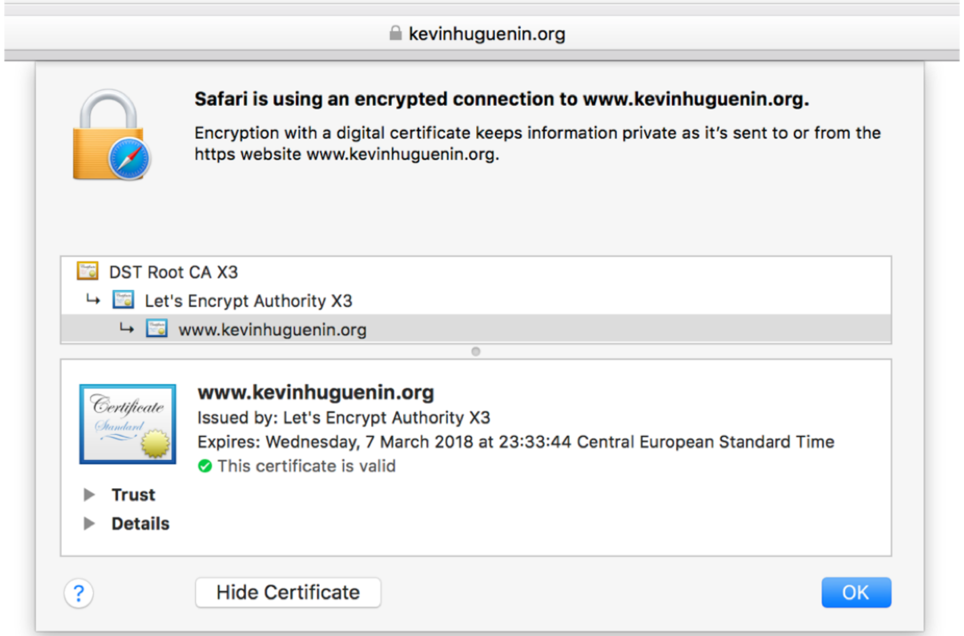
- Making HTTP requests is rather straightforward in Python… (http://docs.python-requests.org/)

```
# -*- coding: utf-8 -*-
import requests

if __name__ == '__main__':

    try:
        params = {'hl': 'en', 'q': 'python'}
        r = requests.get('http://www.google.ch', params=params)
        print('Code:', r.status_code)
        print('Headers:', r.headers)
        print('Text: ', r.text[:130] + '...')
    except Exception as e:
        print(' (exception: {:s}'.format(str(e)))
```

Démo

THE HTTP(S) PROTOCOL

- A few words on HTTPS
  - HTTPS means that the HTTP protocol is run on top of a secure socket (SSL)
  - SSL relies on a digital certificate (see cryptography lecture)
    - Provides cryptographic material (i.e., keys) to process (i.e., encrypt) the data sent in such a way that only the recipient can read (i.e., decrypt) the data
    - Provides a proof of ownership (i.e., signature) of the website issue by a certification authority (chain)



kevinhuguenin.org

Safari is using an encrypted connection to www.kevinhuguenin.org.

Encryption with a digital certificate keeps information private as it's sent to or from the https website www.kevinhuguenin.org.

DST Root CA X3
↳ Let's Encrypt Authority X3
↳ www.kevinhuguenin.org

www.kevinhuguenin.org
Issued by: Let's Encrypt Authority X3
Expires: Wednesday, 7 March 2018 at 23:33:44 Central European Standard Time
✓ This certificate is valid

THE HTTP(S) PROTOCOL

- SSL provides guarantees in terms of
  - Authentication
  - Integrity
  - Confidentiality

- traffic analysis still possible, IPs still visible for eavesdropper (see information security and privacy lecture)

THE HTML FORMAT

- Most web pages returned by web servers are text files in the HTML (Hypertext Markup Language) format

```
<!DOCTYPE html>
<html xmlns:og="http://ogp.me/ns#">
    <head>
        <meta charset="utf-8">
…
```

- HTML loosely derives from the XML (eXtended Markup Language) format
  - Based on markups (opening and closing, possibly with attributes): `<grade>5.5</grade>`
  - Markups are nested (tree structure)

```
   <?xml version="1.0" encoding="UTF-8"  standalone="yes"?>
   <data origin="grade server">
children  <grade>5.5</grade>
          <grade>3.9</grade>
          <grade>2.8</grade>
          <!- end grades->
   </data>
```

THE HTML FORMAT

- Elements of HTML files correspond to graphical elements
  - Titles: `<h1>, <h2>`, *etc.*
  - Images: `<img>`
  - Links: `<a>`
  - Tables: `<table>` (`<tr>` for lines, `<td>` for cells)
- The styles of which can be adjusted by using attributes

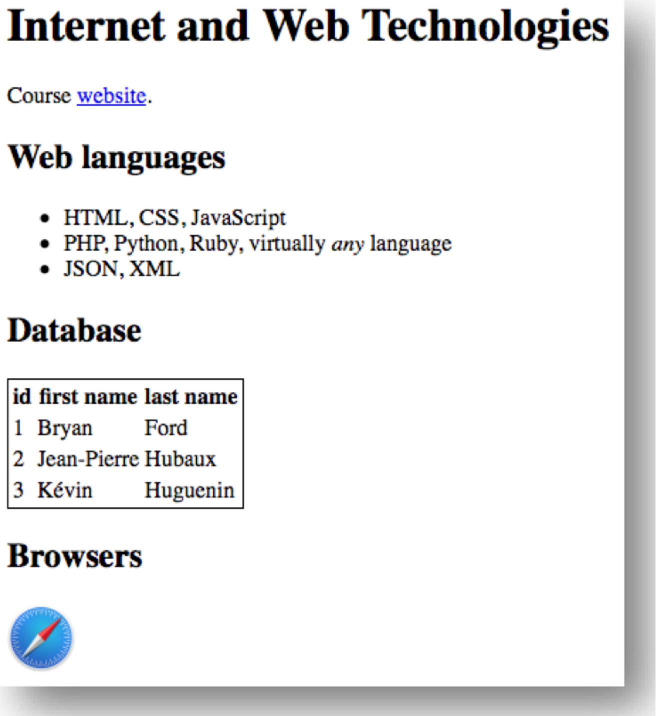
- The web browser takes care of the rendering the web pages, downloading the linked resources (e.g., images; through HTTP requests) and handling user actions (e.g., clicks)

Démo

THE HTML FORMAT: EXAMPLE

Add a line
Add a link
Add a section

```
<html lang="en">
    <head>
        <title>ICRC CS Class</title>
    </head>
    <body>
        <h1>Internet and Web Technologies</h1>
        <p>Course <a href="https://moodle.epfl.ch/course/view.php?id=15667">website</a>.</p>
        <h2>Web languages</h2>
        <ul>
            <li>HTML, CSS, JavaScript</li>
            <li>PHP, Python, Ruby, virtually <i>any</i> language</li>
            <li>JSON, XML</li>
        </ul>
        <h2>Database</h2>
        <table style="border: 1px solid black;">
            <tr>
                <th>id</th>
                <th>first name</th>
                <th>last name</th>
            </tr>
            <tr>
                <td>1</td>
                <td>Bryan</td>
                <td>Ford</td>
            </tr>
            […]
        </table>
        <h2>Browsers</h2>
        <img src="safari.png" alt="Safari's logo" width="48" height="48"/>
    </body>
</html>
```



Internet and Web Technologies

Course website.

Web languages

- HTML, CSS, JavaScript
- PHP, Python, Ruby, virtually *any* language
- JSON, XML

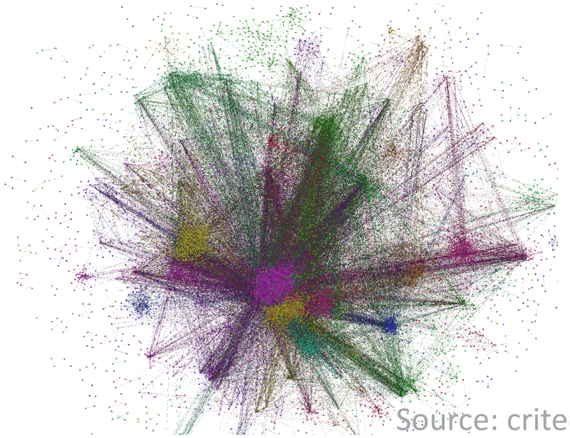
Database

id	first name	last name
1	Bryan	Ford
2	Jean-Pierre	Hubaux
3	Kévin	Huguenin

Browsers

THE WORLD WIDE WEB (WWW)

- Hypertext language (HTML) + transfer protocol (HTTP) + linked resources + Internet: The World Wide Web!
  - The Web forms a directed graph:
    A web page is a node; there is an edge from a page to another if the former contains a link to the latter



Source: criteo

- Invention at CERN in the 80's by Tim Berners-Lee and colleagues
- Web 2.0
  - Internet users contribute to producing web pages (or at least produce content)
- Semantic Web

REFINEMENTS: THE CSS FORMAT

- **Principle**: Separate content (pure XML/HTML) and presentation/formatting/style
- **Solution**: Cascading Style Sheets (CSS)
  - Define styles for each (type of) elements
  - Styles are inherited by children (elements embedded in other elements inherits the style of their parent element)

```
a {
   color: green;
}
li {
   padding: 0.25cm;
}
p {
   font-size: 150%;
}
.important_text {
   font-weight: bold;
}
```

Change font size for section headers
Change text color for table cells

  - Separate file (local or remote), linked from HTML

```
<head>
   <meta charset="UTF-8">
   <title>ICRC CS Class</title>
   <link rel="stylesheet" type="text/css" href="style.css">
</head>
```

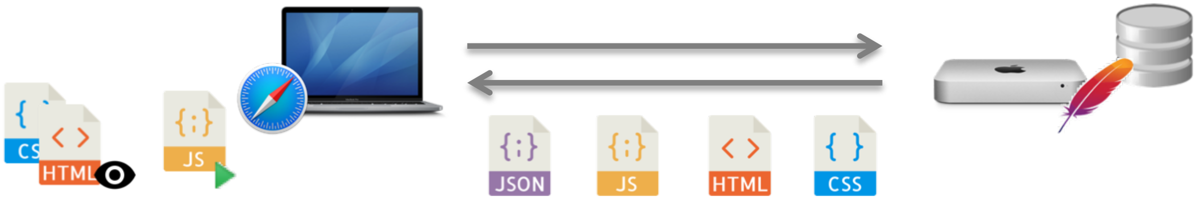
XML JSON + Démo JS

REFINEMENTS: THE JS FORMAT

- **Principle**: Dynamically modify the web pages locally (w/o refreshing the page)
- **Solution**: Scripts embedded in web pages, executed at the client side (by the browser).
  The JavaScript language.
  - Manipulates HTML elements through simple variables
    ```
    var x = getElementById('group_options’);
    ```
  - Add / remove / re-arrange content; change the style
    ```
    <a onclick="getElementById('group_options').style.display = 'none';">hide</a>
    <p id="group_options">Bla bla bla…<p>
    ```
  - Fetch remote content (through HTTP request)—typically in XML or JSON format—and insert it
- Separate file (local or remote), linked from HTML
  ```
  <script type="text/JavaScript" src="main.js"></script>
  ```
- Faster than refreshing the web page from the server
  - Remote program running locally (BitCoin miner; exfiltration of sensitive information, e.g., mouse tracking, screen resolution). (see information security and privacy lecture)

WEB TECHNOLOGIES: SUMMARY

1. Web clients—browsers—make HTTP requests for a URL with some parameters and data (GET, POST)
2. Web servers generate the requested resource (back-end; PHP, Python, Ruby—virtually any language), possibly by fetching data from a database, and sent it back through a HTTP response
   - Webpages: structured data (HTML~XML) plus
     - Style data (CSS)
     - Program to dynamically modify the page without reloading—and possibly make other requests (JS, AJAX)
   - Web service / APIs: structured data such as XML and JSON
3. Web client renders the page from the HTML + CSS code and runs the JS code (front-end)



Full-stack developper (n.): a person who knows (and hopefuly masters) all this

WEB BROWSERS

- Web browsers (software) are an essential piece of the web eco system
  - Fetch remote content
  - Render content
  - Execute scripts that modify the content
  - In a secure and efficient way
- Not all browsers are equal
  - Safari vs. Chrome vs. Firefox; mobile vs. Desktop; versions
  - They identify themselves to the web server, through a so-called user-agent string, so that the server can adapt the content

```
Mozilla/5.0 (iPhone; CPU iPhone OS 10_3_1 like Mac OS X) AppleWebKit/603.1.30 (KHTML, like
Gecko) Version/10.0 Mobile/14E304 Safari/602.1
```

- Tracking! (see information security and privacy lecture)

WEB SITES

- Very few people do write HTML / CSS / JS code manually

- Web developpers and webmasters heavily rely on frameworks
  - Content management service (CMS) such as Jahia, WordPress, Drupal
  - Back-end frameworks such as Django, Ruby on rails, NodeJS
  - Front-end frameworks such as Vue.js, Angular

WEB APIS

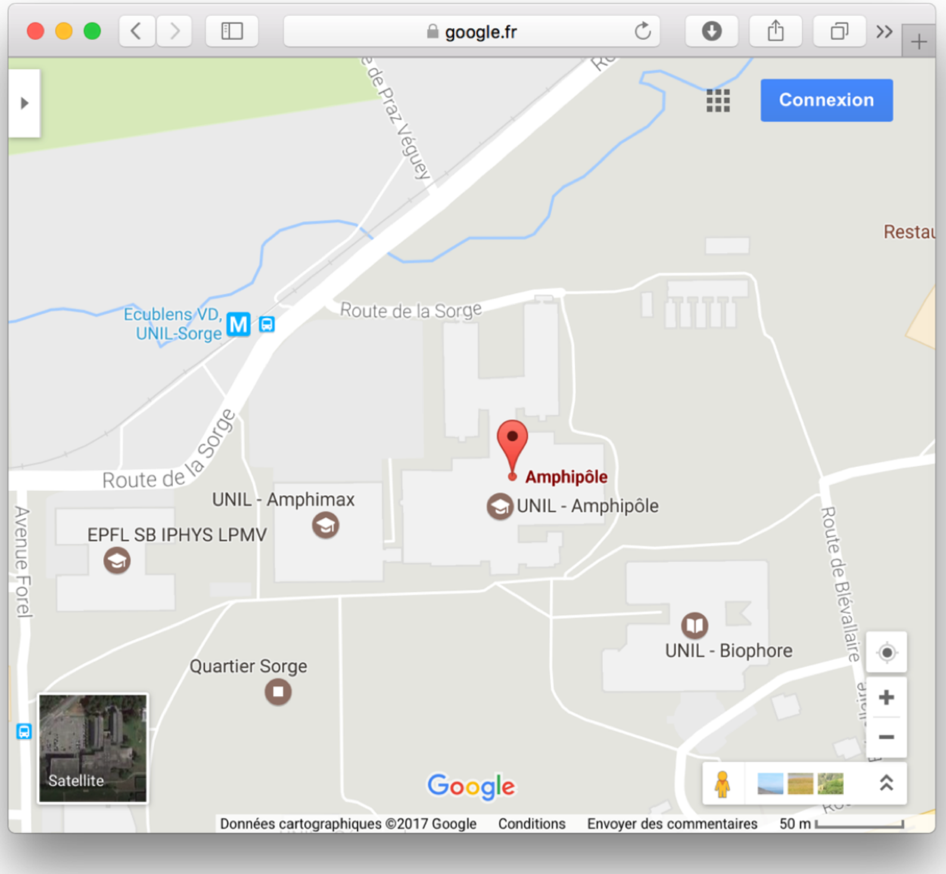
- Originally, web servers were returning only HTML content aimed at being rendered for and displayed to human beings

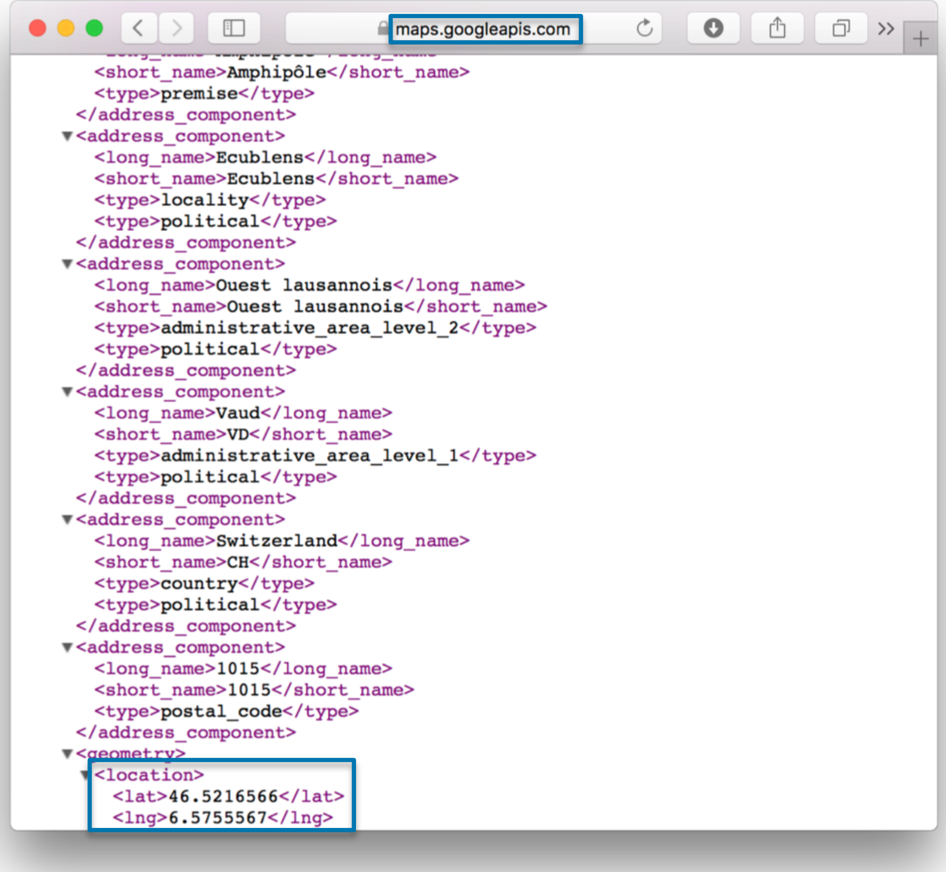
- HTML content is not easy for program to analyze and manipulate (sloppy) and heavyweight (eye candy)

- Nowadays, many web servers provide web services, accessible through an application programming interface (API) over HTTP: a web API.

Demo

WEB APIS





```
    <short_name>Amphipôle</short_name>
    <type>premise</type>
  </address_component>
▼<address_component>
    <long_name>Ecublens</long_name>
    <short_name>Ecublens</short_name>
    <type>locality</type>
    <type>political</type>
  </address_component>
▼<address_component>
    <long_name>Ouest lausannois</long_name>
    <short_name>Ouest lausannois</short_name>
    <type>administrative_area_level_2</type>
    <type>political</type>
  </address_component>
▼<address_component>
    <long_name>Vaud</long_name>
    <short_name>VD</short_name>
    <type>administrative_area_level_1</type>
    <type>political</type>
  </address_component>
▼<address_component>
    <long_name>Switzerland</long_name>
    <short_name>CH</short_name>
    <type>country</type>
    <type>political</type>
  </address_component>
▼<address_component>
    <long_name>1015</long_name>
    <short_name>1015</short_name>
    <type>postal_code</type>
  </address_component>
▼<geometry>
  ▼<location>
      <lat>46.5216566</lat>
      <lng>6.5755567</lng>
```

Web site, HTML, human being

Web API, XML/JSON, program (e.g., JS, Python)

Demo

WEB APIS: EXAMPLES

- Examples
  - Maps : Google Maps (exemple)*, HERE*, *etc*.
  - Criminality : Iowa Sex Offenders Registry (example), Data Police UK (example), *etc*.
  - Social networks: Facebook*,**, Twitter*,**, Foursquare*,**, RunKeeper*,**, *etc*.
  - Medicine : NIH Drugs (example), *etc*.
  - Movies and music: Spotify (example), TMDb (example), *etc*.
  - Scientific publications: HAL archives ouvertes (example), *etc*.

* Requires an account (API key)

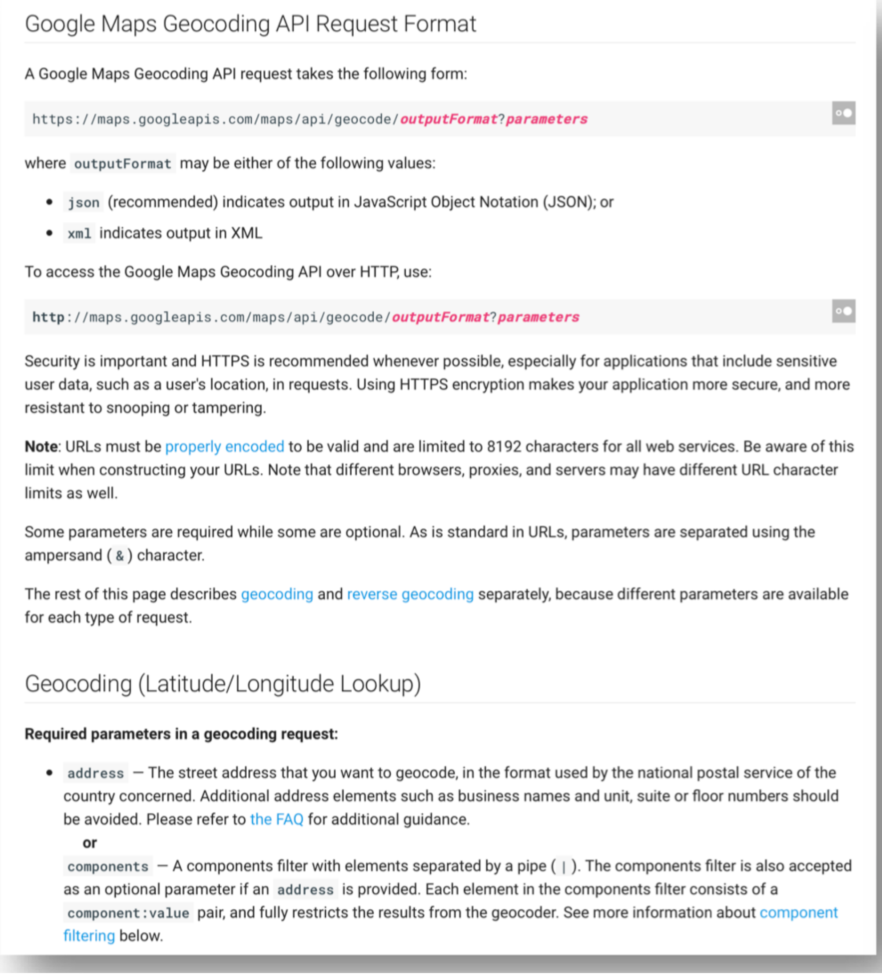
** Requires an authorization token

- Why a Web API instead of a local library with and API?
  - Computation, storage, maintenance
  - Monetization

Demo

WEB APIS

- Web service provide a documentation for their web APIs: Endpoints and Parameters



Google Maps Geocoding API Request Format

A Google Maps Geocoding API request takes the following form:

```
https://maps.googleapis.com/maps/api/geocode/outputFormat?parameters
```

where `outputFormat` may be either of the following values:

- `json` (recommended) indicates output in JavaScript Object Notation (JSON); or
- `xml` indicates output in XML

To access the Google Maps Geocoding API over HTTP, use:

```
http://maps.googleapis.com/maps/api/geocode/outputFormat?parameters
```

Security is important and HTTPS is recommended whenever possible, especially for applications that include sensitive user data, such as a user's location, in requests. Using HTTPS encryption makes your application more secure, and more resistant to snooping or tampering.

Note: URLs must be properly encoded to be valid and are limited to 8192 characters for all web services. Be aware of this limit when constructing your URLs. Note that different browsers, proxies, and servers may have different URL character limits as well.

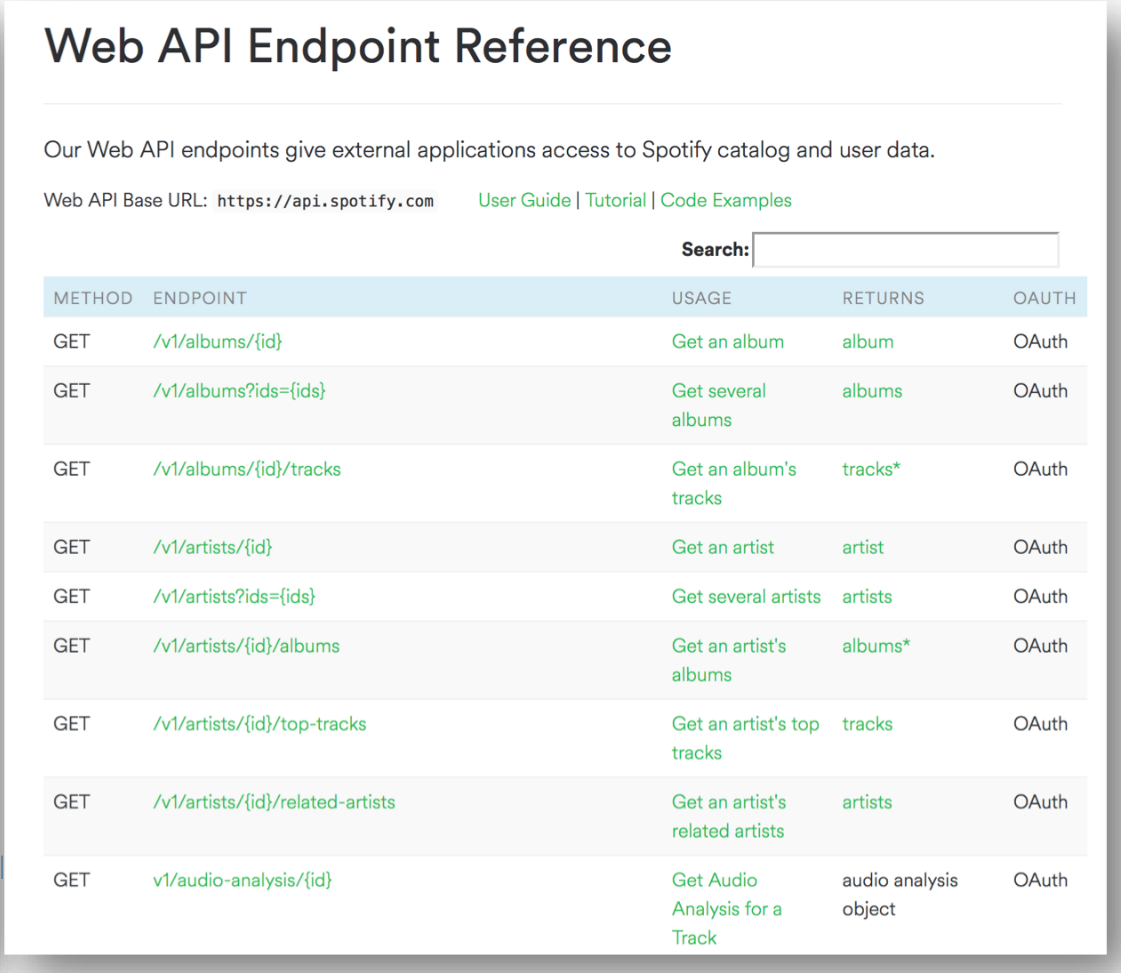
Some parameters are required while some are optional. As is standard in URLs, parameters are separated using the ampersand (`&`) character.

The rest of this page describes geocoding and reverse geocoding separately, because different parameters are available for each type of request.

Geocoding (Latitude/Longitude Lookup)

Required parameters in a geocoding request:

- `address` — The street address that you want to geocode, in the format used by the national postal service of the country concerned. Additional address elements such as business names and unit, suite or floor numbers should be avoided. Please refer to the FAQ for additional guidance.
  or
  `components` — A components filter with elements separated by a pipe (`|`). The components filter is also accepted as an optional parameter if an `address` is provided. Each element in the components filter consists of a `component:value` pair, and fully restricts the results from the geocoder. See more information about component filtering below.



Web API Endpoint Reference

Our Web API endpoints give external applications access to Spotify catalog and user data.

Web API Base URL: `https://api.spotify.com` User Guide | Tutorial | Code Examples

Search:

METHOD	ENDPOINT	USAGE	RETURNS	OAUTH
GET	/v1/albums/{id}	Get an album	album	OAuth
GET	/v1/albums?ids={ids}	Get several albums	albums	OAuth
GET	/v1/albums/{id}/tracks	Get an album's tracks	tracks*	OAuth
GET	/v1/artists/{id}	Get an artist	artist	OAuth
GET	/v1/artists?ids={ids}	Get several artists	artists	OAuth
GET	/v1/artists/{id}/albums	Get an artist's albums	albums*	OAuth
GET	/v1/artists/{id}/top-tracks	Get an artist's top tracks	tracks	OAuth
GET	/v1/artists/{id}/related-artists	Get an artist's related artists	artists	OAuth
GET	v1/audio-analysis/{id}	Get Audio Analysis for a Track	audio analysis object	OAuth

Demo

WEB APIS: PROGRAMMING

```
# -*- coding: utf-8 -*-
import requests
import json

if __name__ == '__main__':

    address = input('Address? ')

    params = {'language': 'fr', 'address': address}
    r = requests.get('https://maps.googleapis.com/maps/api/geocode/json', params=params)

    obj = json.loads(r.text)

    for res in obj['results']:
        print('%s (%.5f, %.5f)' % (res['formatted_address’],
                                   res['geometry']['location']['lat’],
                                   res['geometry']['location']['lng']))
    # epfl
    # croix rouge geneve
```

Demo

WEB APIS

- Web APIs giving access to user data, typically social platforms (e.g., Facebook, Twitter, RunKeeper, Spotify), require a token obtained after the user agrees.
  - Performed through Open Authentication (OAuth)

1. The original website redirects the user to the Twitter website to obtain their agreement
2. The user logs in on Twitter, reviews the accesses requested by the original website and agrees
3. The original website receives an authorization token, which it must attach to all its requests (in the HTTP headers) to the Web API

- Similar to Single-Sign-On (SSO)

Demo

THE DARK SIDE OF THE WEB

- **The invisible web**: Websites that are not indexed and referenced by search engines

- **The dark web**: Websites that cannot be directly located and accessed on the Internet (no direct hostname or IP address, e.g., .onion accessible through TOR—see information security and privacy lecture)
  - Mostly for illegal activities (e.g., the silk road)

SEARCH ENGINES

- Index and reference web pages on the Internet: storing pages in a highly efficient way in order to answer queries
  - Navigate (crawl) the web by following links on web pages

- Enable users to make free-text queries

- Search is complex
  - Speed and relevance are key
  - The web is huge (scalability)
  - Fuzzy matching because of typos

Amazon found every 100ms of latency cost them 1% in sales.

_ The Cost of Latency _

- Examples: Google, DuckDuckGo (privacy-preserving), Exalead (EU initiative)

Adapted from Sylvain Utard's, VP of Engineering at Algolia

SEARCH ENGINES

- Core techniques:
  - Matching (find pages containing the keywords; coarse-grained)
    - Use of inverted lists: for each word, store the list of pages containing this word
    - Use links between similar words (w.r.t. the edit distance, e.g., that differ by only one letter)
  - Ranking (order the returned pages by relevance; fine-grained)
    - Use of metrics such as
      - Term frequency-inverse document frequency (TF-IDF) and
      - PageRank (importance of a web page; how often would a random navigation visit it)

THE BEST PLACE TO HIDE A DEAD BODY
IS PAGE TWO OF THE GOOGLE SEARCH RESULTS

    - Fairness / Business opportunity; Search Engine Optimization (SEO); Google Bombing
- Refinement: personalization, contextualization, query expansion, *etc*.

Adapted from Sylvain Utard's, VP of Engineering at Algolia

THAT'S ALL FOLKS

Questions?

REMINDER

Tomorrow morning, Prof. Falsafi’s lecture on “Cloud and Service Computing” starts at **8:45am** (instead of 9am).