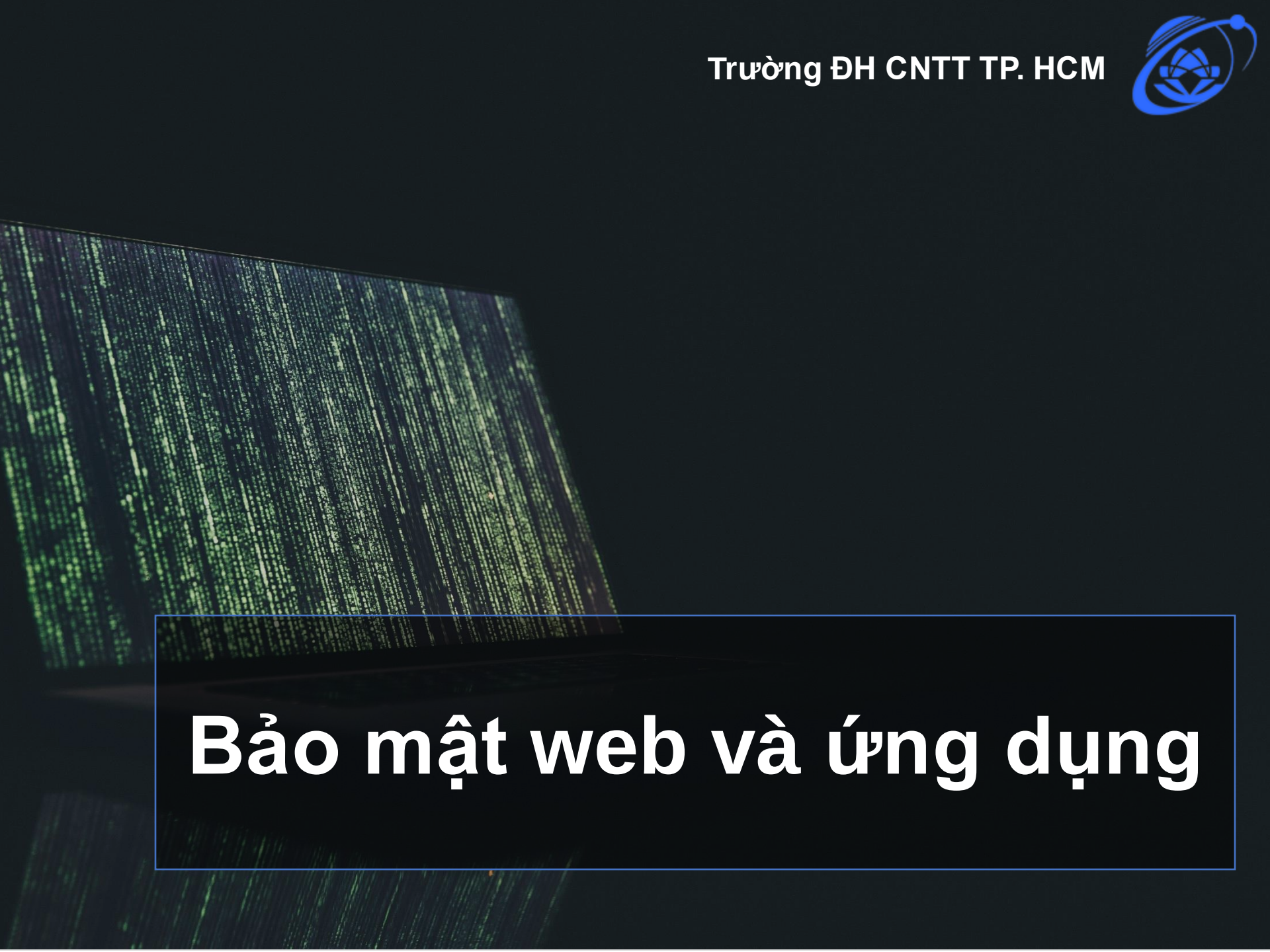
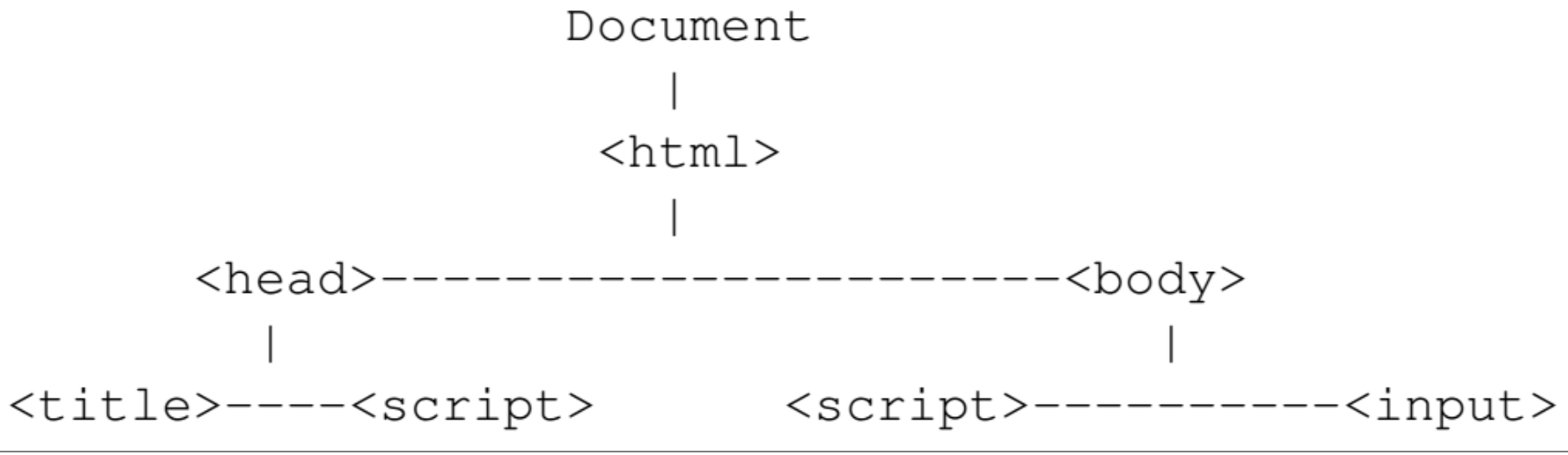
Trường ĐH CNTT TP. HCM



Bảo mật web và ứng dụng

Document Object Model

- Giao diện lập trình độc lập ngôn ngữ và nền tảng cho HTML và XML Document
- Truy cập và thay đổi: content, style, structure



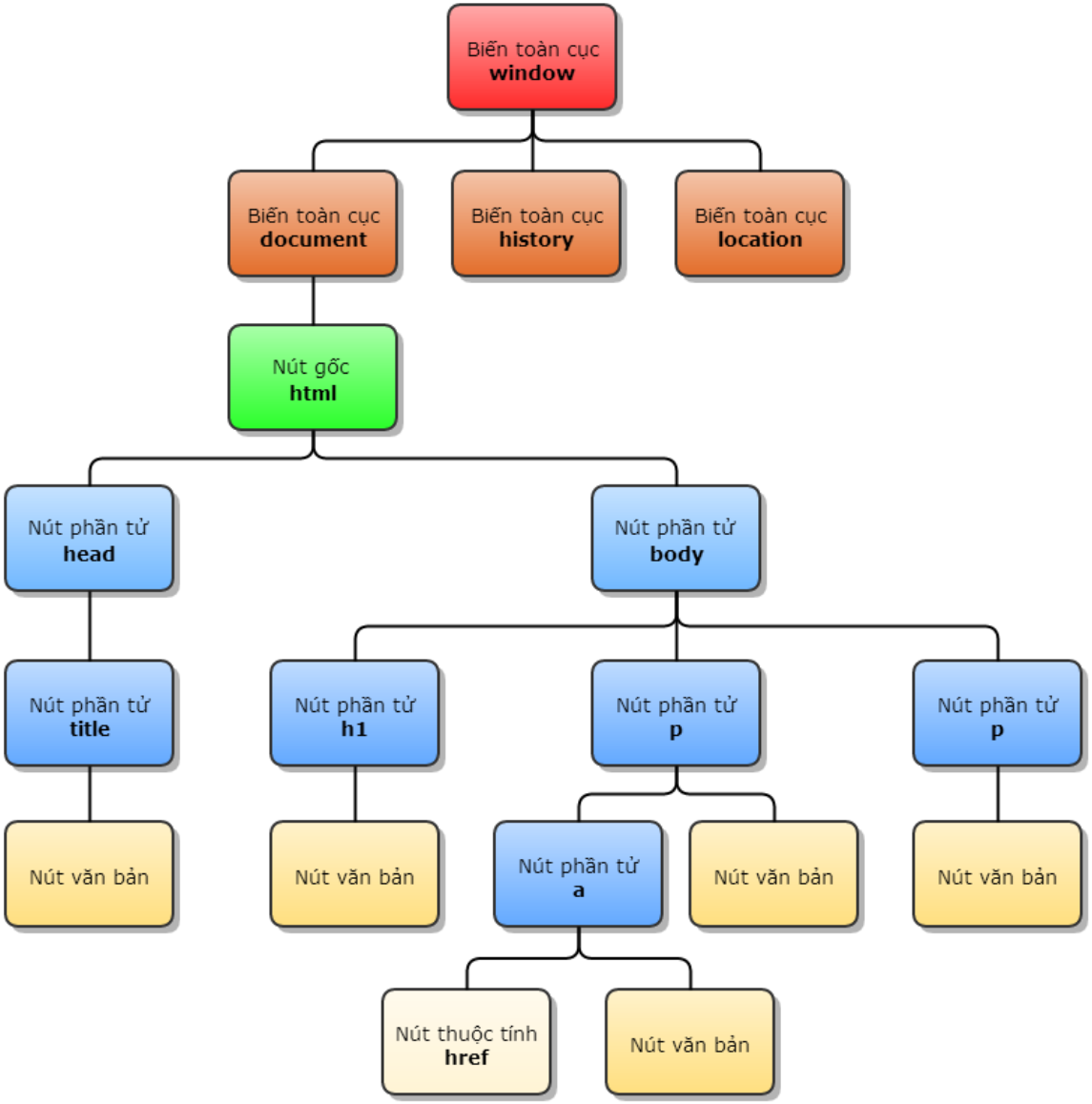
https://www.w3schools.com/js/js_htmldom.asp
https://www.w3.org/DOM/
https://dom.spec.whatwg.org/

HTML DOM

- Mô hình đối tượng chuẩn và giao diện lập trình cho HTML
- Định nghĩa:
  - Thành phần HTML = **object**
  - Property của thành phần HTML
  - Method để truy cập thành phần HTML
  - Events cho thành phần HTML

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

Cấu trúc DOM

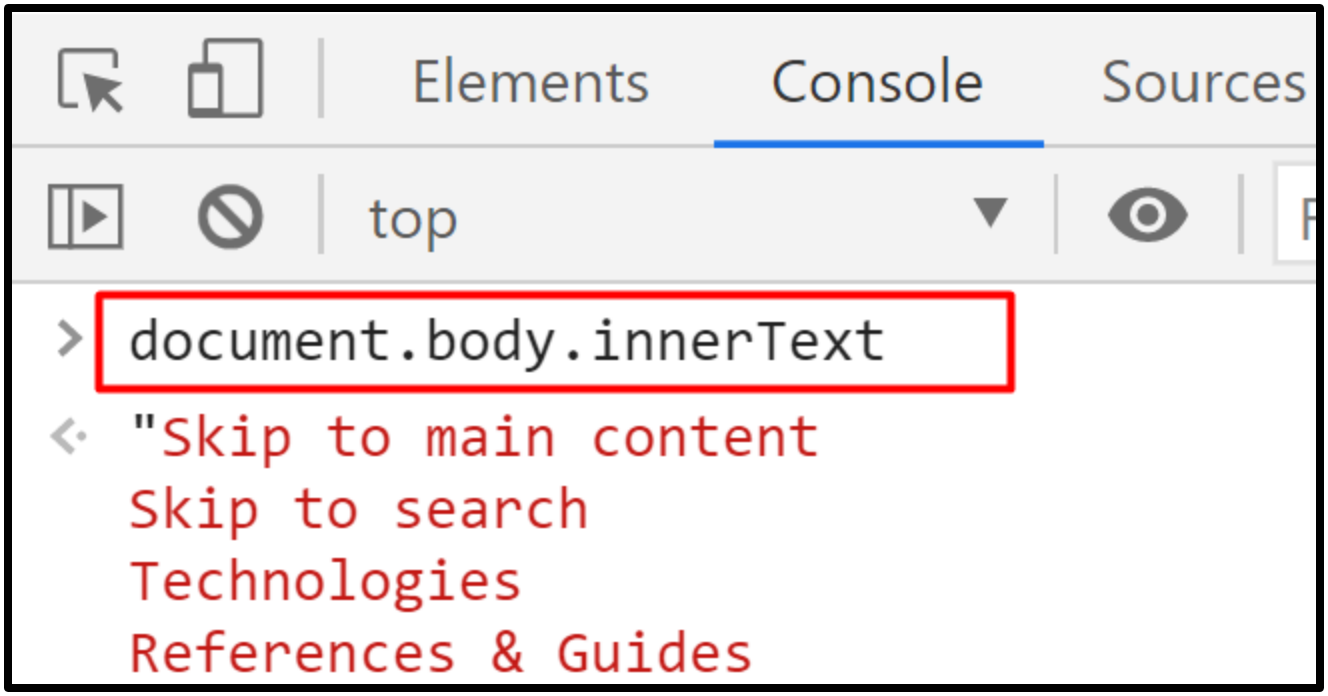

Biến toàn cục
window
Biến toàn cục
document
Biến toàn cục
history
Biến toàn cục
location
Nút gốc
html
Nút phần tử
head
Nút phần tử
body
Nút phần tử
title
Nút phần tử
h1
Nút phần tử
p
Nút phần tử
p
Nút văn bản
Nút văn bản
Nút phần tử
a
Nút văn bản
Nút văn bản
Nút thuộc tính
href
Nút văn bản

Một số thuộc tính phổ biến

id	Định danh – là duy nhất cho mỗi phần tử nên thường được dùng để truy xuất DOM trực tiếp và nhanh chóng.
className	Tên lớp – Cũng dùng để truy xuất trực tiếp như id, nhưng 1 className có thể dùng cho nhiều phần tử.
tagName	Tên thẻ HTML.
innerHTML	Trả về mã HTML bên trong phần tử hiện tại. Đoạn mã HTML này là chuỗi kí tự chứa tất cả phần tử bên trong, bao gồm các nút phần tử và nút văn bản.
outerHTML	Trả về mã HTML của phần tử hiện tại. Nói cách khác, `outerHTML = tagName + innerHTML`.
textContent	Trả về 1 chuỗi kí tự chứa nội dung của tất cả nút văn bản bên trong phần tử hiện tại.
attributes	Tập các thuộc tính như id, name, class, href, title...
style	Tập các thiết lập định dạng của phần tử hiện tại.
value	Lấy giá trị của thành phần được chọn thành một biến.

Một số thuộc tính phổ biến

- nodeValue
- nodeName
- nodeType
- document.body: body của HTML
- document.documentElement: toàn bộ nội dung HTML



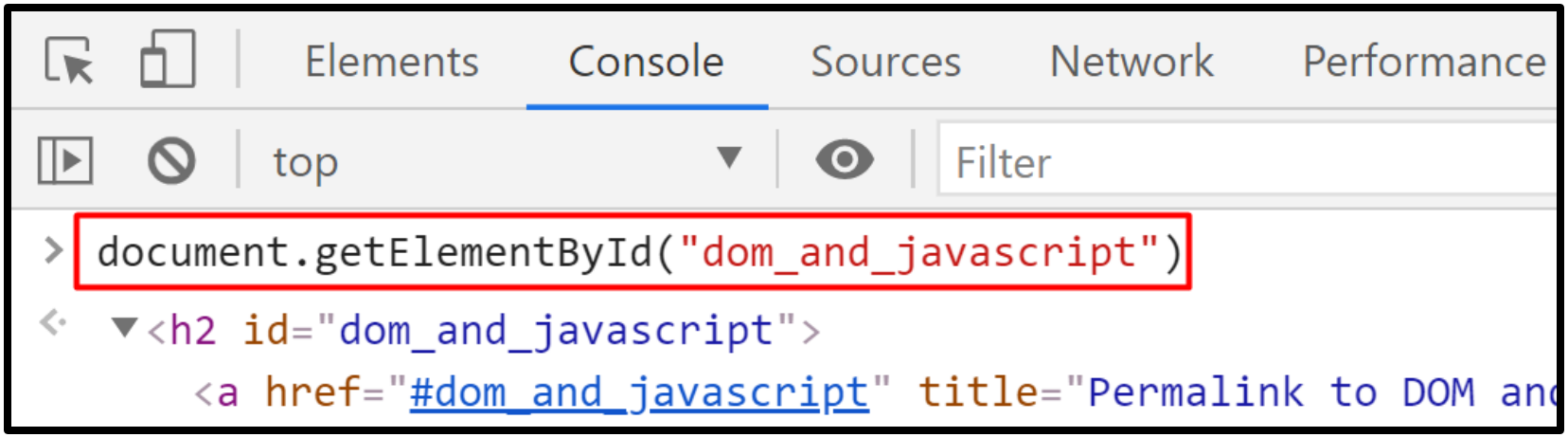
Một số phương thức phổ biến

getElementById(id)	Tham chiếu đến 1 nút duy nhất có thuộc tính `id` giống với id cần tìm.
getElementsByTagName(tagname)	Tham chiếu đến tất cả các nút có thuộc tính `tagName` giống với tên thẻ cần tìm, hay hiểu đơn giản hơn là tìm tất cả các phần tử DOM mang thẻ HTML cùng loại. Nếu muốn truy xuất đến toàn bộ thẻ trong tài liệu HTML thì hãy sử dụng `document.getElementsByTagName('*')`.
getElementsByName(name)	Tham chiếu đến tất cả các nút có thuộc tính `name` cần tìm.
getAttribute(attributeName)	Lấy giá trị của thuộc tính.
setAttribute(attributeName, value)	Sửa giá trị của thuộc tính.
appendChild(node)	Thêm 1 nút con vào nút hiện tại.
removeChild(node)	Xóa 1 nút con khỏi nút hiện tại.

https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_doc_writeln

Thêm / Xóa HTML element

Method	Description
document.createElement(*element*)	Create an HTML element
document.removeChild(*element*)	Remove an HTML element
document.appendChild(*element*)	Add an HTML element
document.replaceChild(*element*)	Replace an HTML element
document.write(*text*)	Write into the HTML output stream



HTML DOM EventListener

- Dùng hàm **addEventListener()** để thêm event handler cho HTML element
  - Không viết đè handler đang có
  - Thêm nhiều handler cho 1 element
  - Thêm cho bất kỳ đối tượng DOM
  - Tách biệt HTML → dễ đọc
  - Dễ dàng remove event listener dùng **removeEventListener()**
- Cú pháp:

  element.addEventListener(event, function, useCapture);

HTML DOM EventListener

- Ví dụ:

1. ```
   element.addEventListener("click", myFunction);
   ```

2. ```
   element.addEventListener("click", function(){ alert("Hello World!"); });
   ```

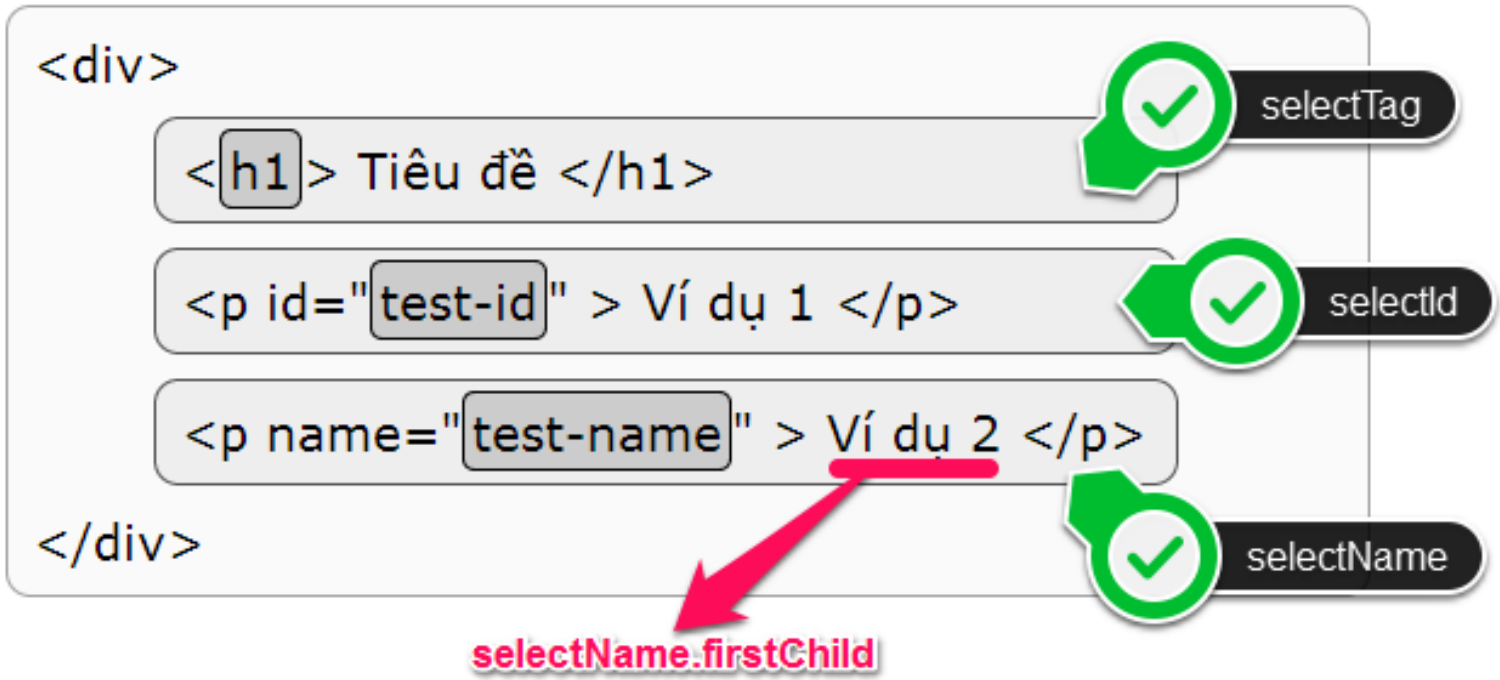
3. ```
   element.addEventListener("click", myFunction);

   function myFunction() {
       alert ("Hello World!");
   }
   ```

4. ```
   window.addEventListener("resize", function(){
       document.getElementById("demo").innerHTML = sometext;
   });
   ```

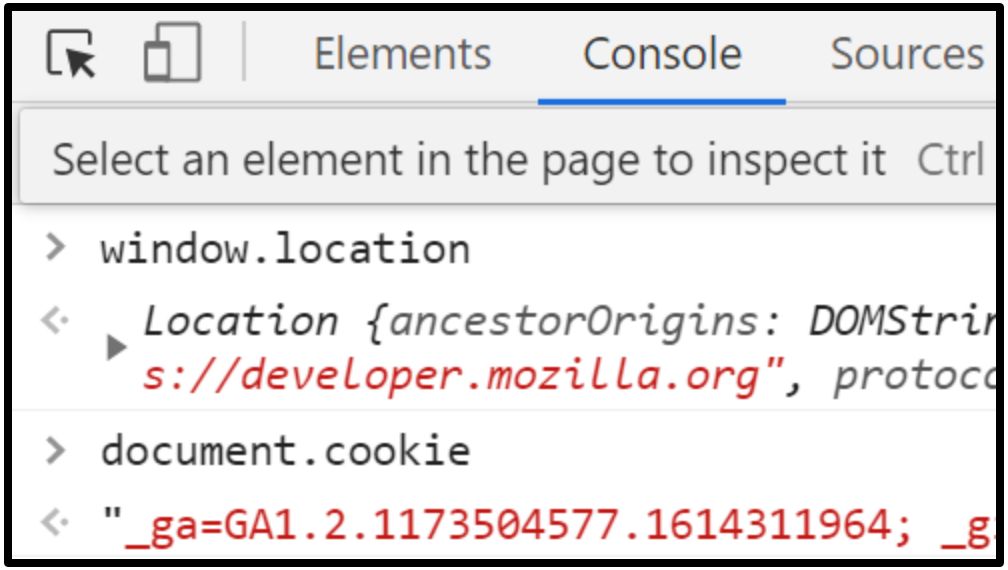
Truy xuất DOM

```
var selectId = document.getElementById('test-id');
var selectName = document.getElementsByName('test-name')[0];
var selectTag = document.getElementsByTagName('h1')[0];
```



Browser Object Model

- Cho phép JS “nói chuyện” với trình duyệt
- Đối tượng window:
  - window.location
  - window.history
  - window.navigator
  - Timing Events
    - setTimeout()
    - setInterval()
  - document.cookie

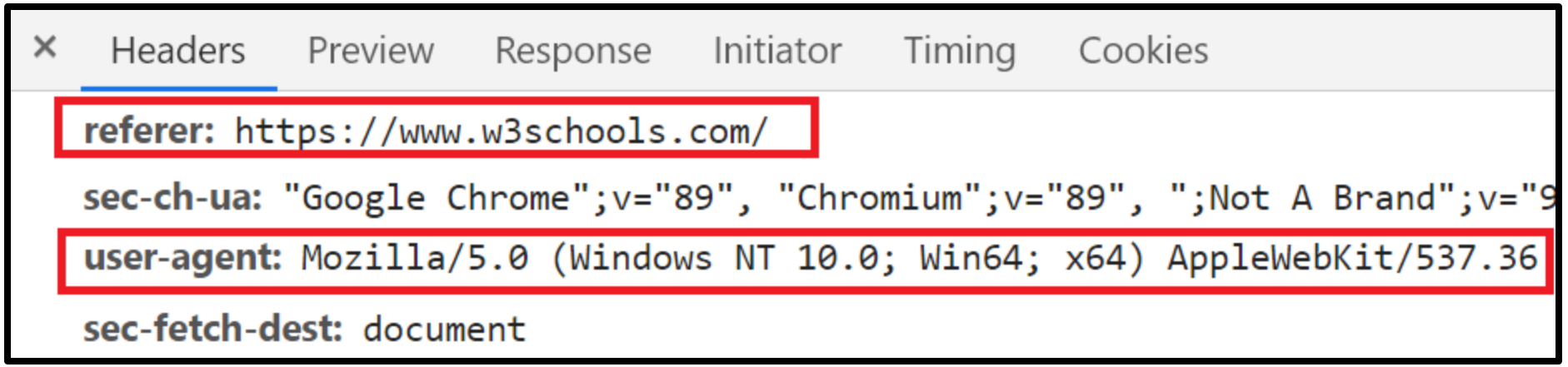


Nhận diện HTTP Client/Session

- HTTP không duy trì trạng thái
- Thông tin trạng thái có thể truyền bằng cách sử dụng:
  - HTTP Headers
  - Địa chỉ IP của Client
  - Đăng nhập của người dùng
  - FAT URLs
  - Cookies

HTTP Header

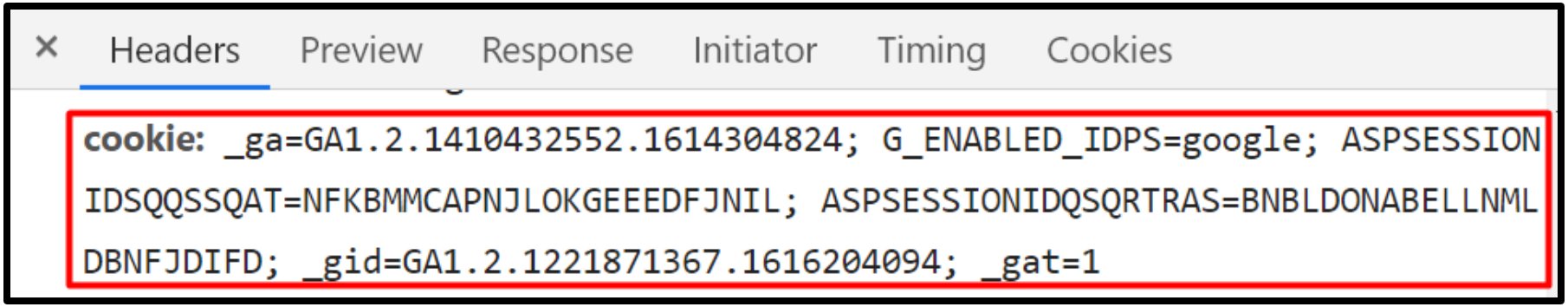
- From
  - Địa chỉ email của người dung trong (HTTP request).
- User-Agent
  - Phiên bản trình duyệt của người dung (HTTP request)
- Referer
  - Trang đã đưa người dùng đến đường dẫn



HTTP Header

- Authorization
  - Username & Password.
- Client-ip
- X-Forwarded-For
- Cookie

HTTP Headers - struct	
Accept	text/html, */*; q=0.01
Accept-Charset	ISO-8859-1,utf-8;q=0.7,*;q=0.7
Accept-Encoding	gzip,deflate
Accept-Language	en-us,en;q=0.5
Authorization	Basic dXNlcm5hbWU6cGFzc3dvcmQ=
Connection	keep-alive



Same Origin Policy

- Cơ chế bảo mật quan trọng trong trình duyệt
- Hạn chế *document* hoặc *script* được tải từ một nguồn khác có thể tương tác với tài nguyên từ nguồn hiện tại
- **Mục đích:**

  Bảo vệ người dùng khi truy cập trang web độc hại

https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy
https://www.w3.org/Security/wiki/Same_Origin_Policy

SOP

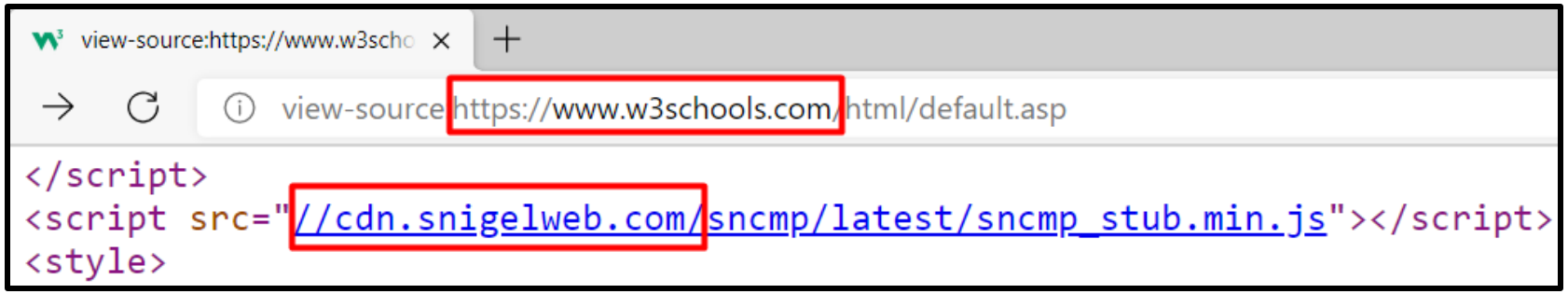
Origin = { ***scheme*** (http, https...)
host
port

Cho URL: http://store.company.com/dir/page.html, đánh giá kết quả là **Same origin** hay không **(Yes/No)**?

URL	Kết quả
http://store.company.com/dir2/other.html	
http://store.company.com/dir/inner/another.html	
https://store.company.com/secure.html	
http://store.company.com:8080/secure.html	
http://shop.company.com/secure.html	

SOP

- Thành phần HTML truy cập cross-origin
  - Các file JavaScript, CSS, frame, iframe,…

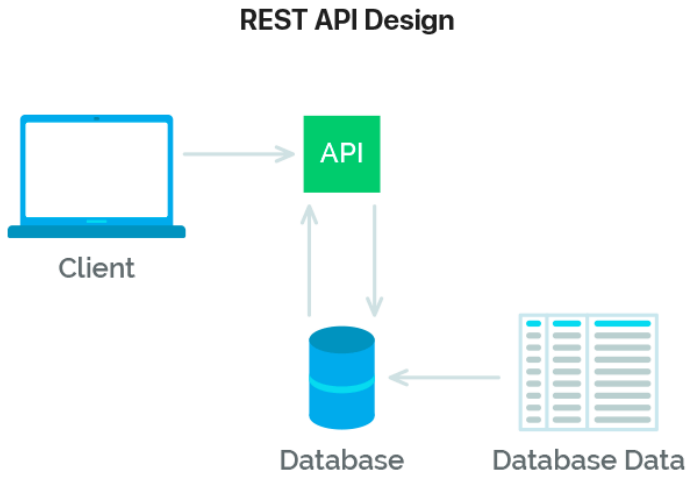


- Cách bypass SOP
  - Thay đổi domain với **document.domain** = “company.com”
  - Sử dụng **CORS** được tích hợp trong HTML5

RESTful API

REST (**RE**presentational **S**tate **T**ransfer) là một chuẩn thiết kế phần mềm web service, nó quy định **cách mà client và server sẽ tương tác với nhau**. Nó đáp ứng những yếu tố sau:

- **Self-documenting** (nhìn vào API ta có thể đoán ra được nó dùng để làm gì)
- **Flexible** (tính mở rộng cũng như tuỳ biến của API)
- **Unified structure and attribute names** (thống nhất về mặt cấu trúc cho resource cũng như cách đặt tên cho các attribute)
- **Clear error message** (khi hệ thống xảy ra lỗi thì message phải rõ ràng và chi tiết để phục vụ cho quá trình fix bug)



RESTful API

```
➜  ~ curl https://api.github.com/orgs/MLSDev
```


```
{
  "login": "MLSDev",
  "id": 1436035,
  "url": "https://api.github.com/orgs/MLSDev",
  "repos_url": "https://api.github.com/orgs/MLSDev/repos",
  "events_url": "https://api.github.com/orgs/MLSDev/events",
  "hooks_url": "https://api.github.com/orgs/MLSDev/hooks",
  "issues_url": "https://api.github.com/orgs/MLSDev/issues",
  "members_url": "https://api.github.com/orgs/MLSDev/members{/member}",
  "public_members_url": "https://api.github.com/orgs/MLSDev/public_members{/member}",
  "avatar_url": "https://avatars.githubusercontent.com/u/1436035?v=3",
  "description": "",
  "name": "MLSDev",
  "company": null,
  "blog": "http://mlsdev.com/",
  "location": "Ukraine",
  "email": "hello@mlsdev.com",
  "public_repos": 25,
  "public_gists": 0,
  "followers": 0,
  "following": 0,
  "html_url": "https://github.com/MLSDev",
  "created_at": "2012-02-14T08:35:16Z",
  "updated_at": "2015-10-29T13:48:13Z",
  "type": "Organization"
}
```

RESTful API: HTTP Code phổ biến

- **200 — OK** — Everything is working
- **304 — Not Modified** — The client can use cached data
- **400 — Bad Request** — The request was invalid or cannot be served. The exact error should be explained in the error payload. E.g. „The JSON is not valid“
- **401 — Unauthorized** — The request requires an user authentication
- **403 — Forbidden** — The server understood the request, but is refusing it or the access is not allowed.
- **404 — Not found** — There is no resource behind the URI.
- **422 — Unprocessable Entity** — Should be used if the server cannot process the enitity, e.g. if an image cannot be formatted or mandatory fields are missing in the payload.
- **500 — Internal Server Error** — API developers should avoid this error. If an error occurs in the global catch blog, the stracktrace should be logged and not returned as response.

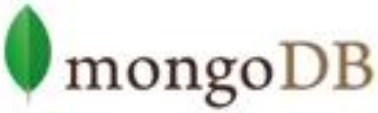
NoSQL


Google Bigtable


riak


CouchDB
relax


redis


mongoDB


Neo4j
the graph database


Cassandra


HBASE


CouchBase


amazon
web services™


Kyoto Cabinet
- lightweight database library -

Các kiểu dữ liệu

- Dữ liệu có thể được chia làm 4 loại:
  1. Dữ liệu có cấu trúc:
     - Có một định dạng được định Nghĩa trước để tổ chức các dữ liệu thành một dạng để dễ lưu trữ, xử lý, truy vấn và quản lý
     - Ví dụ: dữ liệu quan hệ
  2. Dữ liệu không có cấu trúc:
     - Ngược với dữ liệu có cấu trúc
     - Ví dụ: các file nhị phân chứa văn bản, video hoặc audio
     - Lưu ý: dữ liệu không phải hoàn toàn không có cấu trúc (ví dụ, một file audio vẫn có thể có cấu trúc nén và một số siêu dữ liệu metadata kèm theo)

Các kiểu dữ liệu

- Dữ liệu có thể được chia làm 4 loại:
  3. Dữ liệu động:
     - Dữ liệu hay đổi thường xuyên
     - Ví dụ: các file tài liệu văn phòng hoặc các entry giao dịch trong cơ sở dữ liệu tài chính.
  4. Dữ liệu tĩnh:
     - Ngược với dữ liệu động
     - Ví dụ: Dữ liệu hình ảnh y tế như ảnh scan CT hoặc MRI

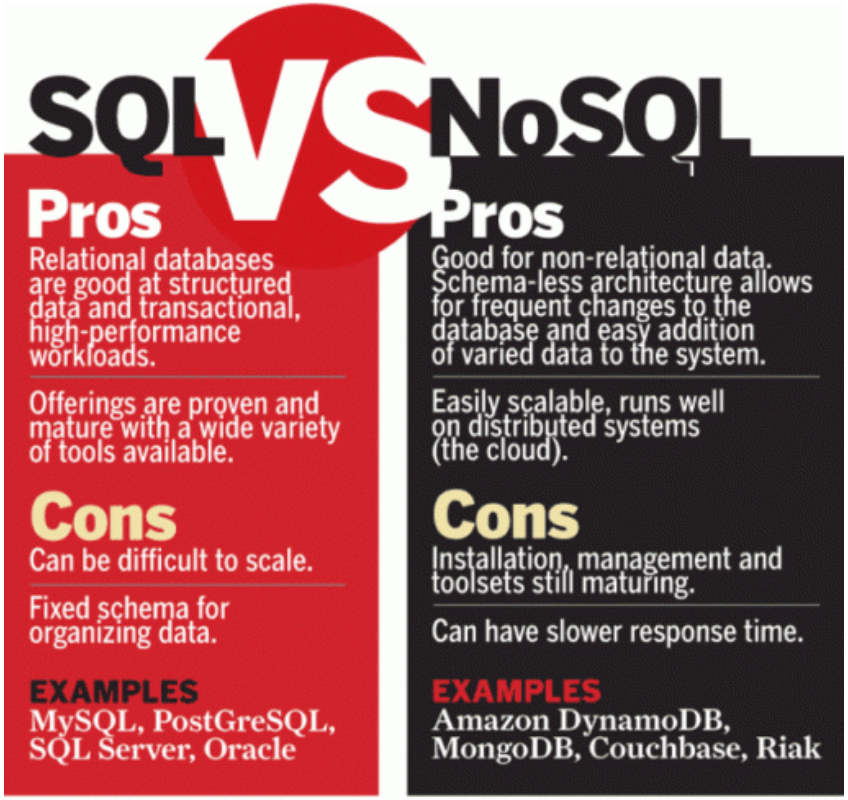
Định lý CAP

- Các ràng buộc của các cơ sở dữ liệu phân tán có thể được mô tả trong **Định lý**
  - **C**onsistency – Tính nhất quán: tất cả các node phải có dữ liệu đồng nhất với nhau (i.e., tính nhất quán chặt chẽ - strict consistency)
  - **A**vailability – Tính sẵn sàng: hệ thống có thể tiếp tục vận hành, thậm chí nếu các node trong cluster bị crash, hoặc một phần phần cứng hay phần mềm bị lỗi do cập nhật
  - **P**artition Tolerance – Khả năng chịu lỗi do phân vùng: hệ thống vẫn có thể vận hành ngay cả khi có phân vùng mạng

Định lý CAP: bất kỳ cơ sở dữ liệu phân tán với dữ liệu chia sẻ nào có nhiều nhất 2 trong 3 thuộc tính mong muốn: C, A hoặc P

NoSQL

- NoSQL bỏ qua tính toàn vẹn của dữ liệu và transaction để đổi lấy hiệu suất nhanh và khả năng mở rộng (scalability).
- Với những ưu điểm trên, NoSQL đang được sử dụng nhiều trong các dự án Big Data, các dự án Real-time, số lượng dữ liệu nhiều.



NoSQL: 4 loại

- Key-Value Database
- Document Database
- Column-Family Database
- Graph Database

NoSQL: **Document Database**

Cơ sở dữ liệu quan hệ

ID	first_name	last_name	year_of_birth
1	'Mary'	'Jones'	1986

ID	user_id	profession
10	1	'Developer'
11	1	'Engineer'

ID	user_id	name	version
20	1	'MyApp'	1.0.4

Cơ sở dữ liệu hướng document

```
{
        first_name: "Mary",
        last_name: "Jones",
        year_of_birth: 1986,
        profession: [  "Developer",
                       "Engineer"
                    ],
        apps: [
              {
                    name: "MyApp",
                    version: 1.0.4
              }
        ]
}
```

Database tiêu biểu: MongoDB, RavenDB, CouchDB, TerraStone, OrientDB

NoSQL: MongoDB (ví dụ)


```
{
  "address": {
     "building": "1007",
     "coord": [ -73.856077, 40.848447 ],
     "street": "Morris Park Ave",
     "zipcode": "10462"
  },
  "borough": "Bronx",
  "cuisine": "Bakery",
  "grades": [
     { "date": { "$date": 1393804800000 }, "grade": "A", "score": 2 },
     { "date": { "$date": 1378857600000 }, "grade": "A", "score": 6 },
     { "date": { "$date": 1358985600000 }, "grade": "A", "score": 10 },
     { "date": { "$date": 1322006400000 }, "grade": "A", "score": 9 },
     { "date": { "$date": 1299715200000 }, "grade": "B", "score": 14 }
  ],
  "name": "Morris Park Bake Shop",
  "restaurant_id": "30075445"
}
```

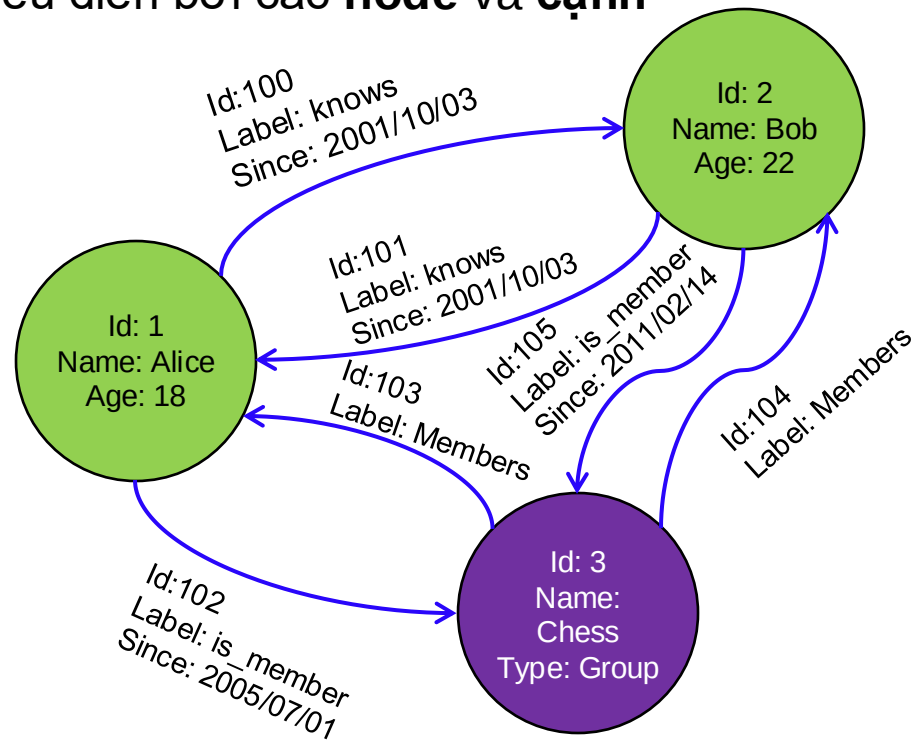

- https://www.w3resource.com/mongodb-exercises/

NoSQL: **Graph Database**

- Dữ liệu trong **graph database (CSDL đồ thị)** được lưu dưới dạng các node. Mỗi node sẽ có 1 số properties như một row trong SQL.
- Các node này được kết nối với nhau bằng các ***relationship***.
- Graph database tập trung nhiều vào relationship giữa các node, áp dụng nhiều thuật toán duyệt node để tăng tốc độ.
- **Database tiêu biểu**: Neo4j, InfiniteGraph, OrientDB, HYPERGRAPHDB

NoSQL: **Graph Database**

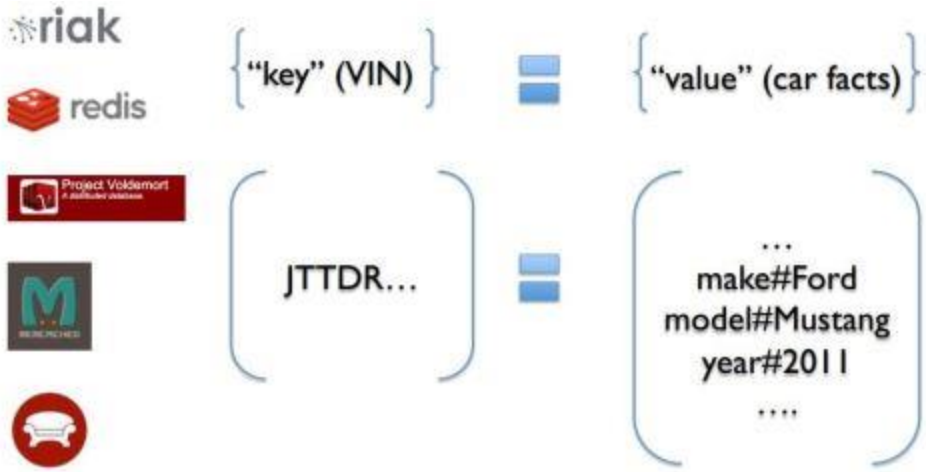
- Dữ liệu được biểu diễn bởi các **node** và **cạnh**



- Graph databases hỗ trợ rất tốt các truy vấn dạng đồ thị (ví dụ, tìm đường đi ngắn nhất giữa 2 thành phần)
- Tiêu biểu: Neo4j and VertexDB

NoSQL: **Key-Value Database**

- Dữ liệu được lưu trữ trong database **dưới dạng key-value**. Để truy vấn dữ liệu trong database, ta **dựa vào key để lấy value**. Các database dạng này có tốc độ truy vấn rất nhanh.
- Database tiêu biểu: Riak, Redis, MemCache, Project Voldemort, CouchBase



NoSQL: **Key-Value Database**

- **Ứng dụng:**

Do tốc độ truy xuất nhanh, key-value database thường được dùng để **làm cache cho ứng dụng** (Tiêu biểu là Redis và MemCache)

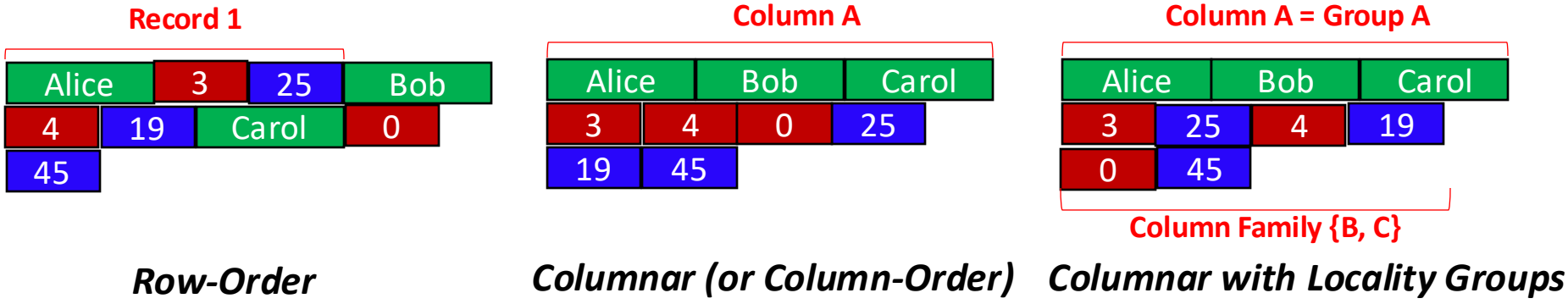
Ngoài ra, nó còn được dùng để lưu thông tin trong sessions, profiles/preferences của user…

NoSQL: **Column-Family Database**

- **Dữ liệu** được lưu trong database **dưới dạng các cột**, thay vì các hàng như SQL. **Mỗi hàng sẽ có một key/id riêng**.
- Điểm đặc biệt là **các hàng trong một bảng sẽ có số lượng cột khác nhau**.
- Câu lệnh truy vấn của nó khá giống SQL.
- CSDL phổ biến: **Cassandra** (Phát triển bởi Facebook), **HyperTable**, Apache **HBase**

NoSQL: **Column-Family Database**

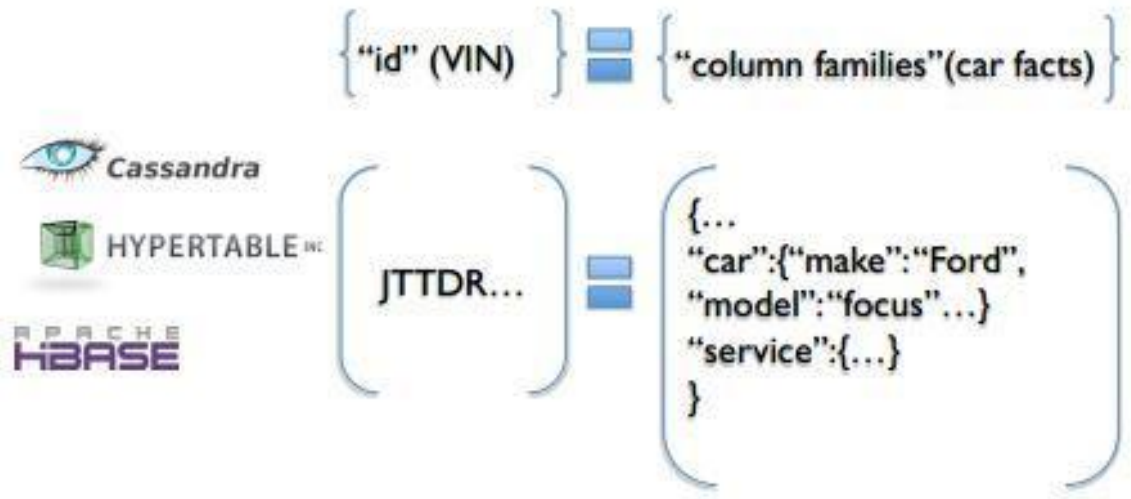
- Columnar databases là dạng lai giữa RDBMSs và Key-Value
  - Values are stored in groups of zero or more columns, but in Column-Order (as opposed to Row-Order)



  - Values are queried by matching keys
- E.g., HBase and Vertica

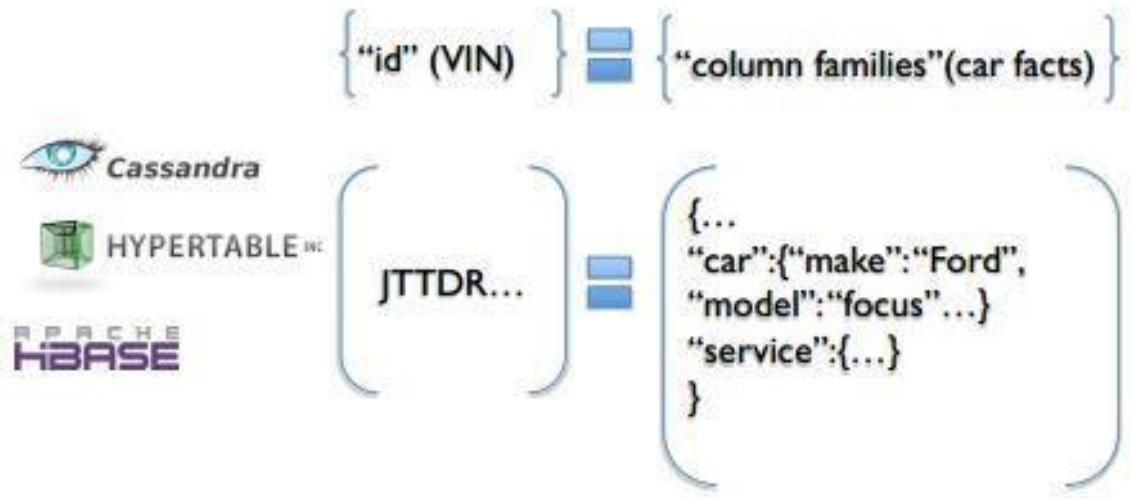
NoSQL: **Column-Family Database**

- Column-Family Database được sử dụng khi ta cần **ghi một số lượng lớn dữ liệu, big data**.
- Nó còn được ứng dụng trong 1 số CMS và ứng dụng e-commerce.



NoSQL: **Column-Family Database**

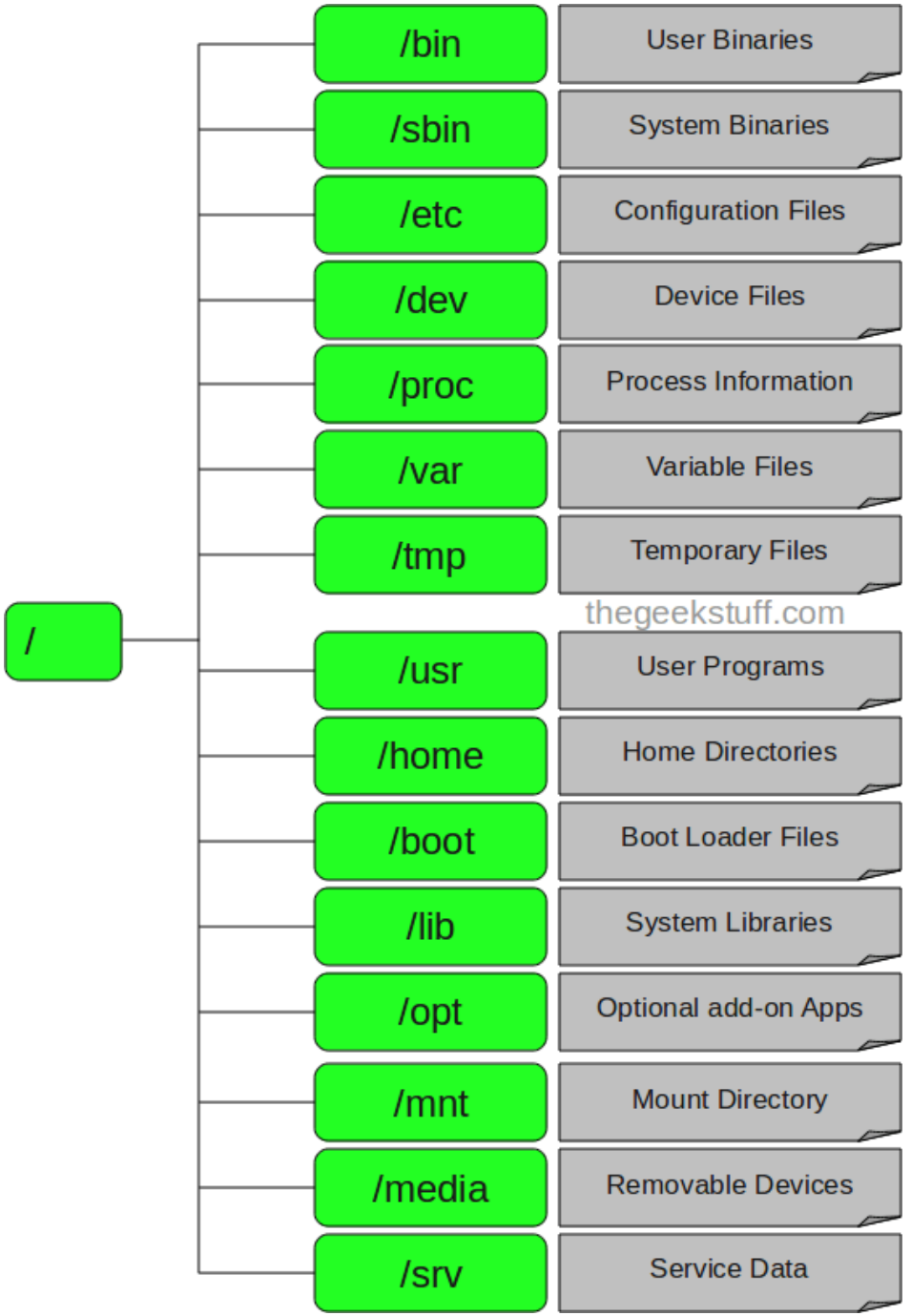
- Column-Family Database được sử dụng khi ta cần **ghi một số lượng lớn dữ liệu, big data**.
- Nó còn được ứng dụng trong 1 số CMS và ứng dụng e-commerce.



MongoDB Code Example in a PHP Application

```php
$username = $_POST['username'];
$password = $_POST['password'];
$connection = new MongoDB\Client('mongodb://localhost:27017');
if($connection) {
        $db = $connection->test;
        $users = $db->users;
        $query = array(
                "user" => $username,
                "password" => $password
        );
        $req = $users->findOne($query);
}
```

Hệ thống tập tin trên Linux



Các lệnh cơ bản trong Linux

- pwd
- cd
- ls
- cat
- cp
- mv
- mkdir
- rm
- find
- grep
- ps
- sudo
- head
- tail
- more
- tar
- chmod
- chown
- kill
- ping
- wget
- top
- echo
- uname
- zip/unzip

- . *là thư mục hiện hành*
- .. *là thư mục cha của thư mục hiện hành*
- ~ *là thư mục gốc của user*

Netcat

- Dùng thiết lập kết nối TCP và UDP
- Gửi nhận file và lệnh

https://quantrimang.com/huong-dan-su-dung-netcat-129

https://www.digitalocean.com/community/tutorials/how-to-use-netcat-to-establish-and-test-tcp-and-udp-connections-on-a-vps

Bảo mật web và ứng dụng

Trường ĐH CNTT TP. HCM