



THỰC HỌC – THỰC NGHIỆP

LẬP TRÌNH JAVASCRIPT

Document Object Model (DOM)

MỤC TIÊU BÀI HỌC

- Lập trình hướng đối tượng trong Javascript
 - Phương thức lập trình
 - Phương thức lập trình hướng đối tượng: đối tượng, thuộc tính và phương thức
 - Mối liên hệ giữa object và mảng
- Browser Object Model



PHẦN I

OOP trong Javascript

Phương thức lập trình

- Lập trình là để giải quyết các vấn đề trong cuộc sống
 - Bài toán tính toán phức tạp: lập trình cho tên lửa bay vào vũ trụ
 - Bài toán logic: đưa ra quyết định (dự báo thời tiết)
 - Bài toán quản lý trong các doanh nghiệp (phần mềm tính lương)



- Phương thức lập trình (programming paradigm) đặc tả cách thức giải quyết vấn đề

Phương thức lập trình

- Có hơn 25 phương thức lập trình
 - Mỗi phương thức lập trình giải quyết cho một vấn đề
 - Một số phương thức khó áp dụng trong thực tiễn lập trình
 - Một số phương thức lập trình chỉ được hưởng ứng bởi một nhóm người hay trong một thời gian ngắn
- Những phương thức lập trình phổ biến
 - Lập trình hướng sự kiện
 - Lập trình hướng thành phần
 - Lập trình cấu trúc
 - Lập trình hướng đối tượng
- Phương thức lập trình hướng đối tượng được phát triển rộng rãi hơn cả

Lập trình hướng đối tượng

- Đối tượng (object): được sử dụng để mô tả các đối tượng ngoài đời thực (các đồ vật, sự vật)
 - Ví dụ đối tượng: quả bóng, cái bàn, ô tô, bông hoa, con người, nhà máy,...
- Mỗi đối tượng có đặc tính và hành động riêng
- Ý tưởng chủ đạo của phương thức lập trình hướng đối tượng: mô phỏng cuộc sống thực trong lập trình
 - Trong cuộc sống có những đối tượng như quả bóng, cái bàn...với các đặc tính và hành động riêng thì lập trình mô phỏng các đối tượng đó với các đặc tính và hành động như thế

Đối tượng (Object)

- Trong lập trình: đặc tính được gọi là thuộc tính, hành động được gọi là phương thức



Đặc tính:

Cân nặng: 2,4kg

Màu lông: nâu

Tuổi: 3

Giống: Chihuahua

Phương thức: Sữa, Cắn



Đặc tính:

Cân nặng: 2 tấn

Màu da: nâu

Vòi: 1m

Phương thức: Phun nước, Ăn cỏ

Lớp (Class)

- Các đối tượng có cùng thuộc tính và phương thức được gom lại thành 1 lớp (Class)
- Hay: Lớp (Class) định nghĩa tập hợp các đối tượng có cùng thuộc tính và phương thức



Đối tượng (Object)

- Tạo đối tượng: sử dụng cặp ngoặc nhọn { }
- Ví dụ

```
let dog = {  
  dogName: "JavaScript",  
  weight: 2.4,  
  color: "brown",  
  breed: "chihuahua",  
  age: 3  
};
```

Đối tượng (Object)

- Truy xuất thuộc tính của đối tượng

```
let dogColor = dog["color"];
```

- hoặc

```
let dogColor = dog.color;
```

- Cập nhật giá trị của thuộc tính của đối tượng

```
dog["color"] = "black";  
dog.weight = 2.3;
```

Đối tượng (Object)

- Phương thức

```
let dog = {  
  dogName: "JavaScript",  
  weight: 2.4,  
  color: "brown",  
  breed: "chihuahua",  
  age: 3,  
  sua: function () { //tenphuongthuc: function(){}  
    return "Gau Gau";  
  }  
};
```

- Truy cập phương thức `dog.sua() //tendoituong.tenphuongthuc()`

Đối tượng (Object)

- Vấn đề nảy sinh

```
let dogtype1 = {  
  weight: 2.4,  
  color: "white",  
  breed: "chihuahua",  
  type: "short hair",  
};
```



```
let dogtype2 = {  
  weight: 2.4,  
  color: "white",  
  breed: "chihuahua",  
  type: "long hair",  
};
```

```
let dogtype3 = {  
  weight: 2.3,  
  color: "brown",  
  breed: "chihuahua",  
  type: "short hair",  
};
```

```
let dogtype4 = {  
  weight: 2.3,  
  color: "black and white",  
  breed: "chihuahua",  
  type: "short hair",  
};
```

- => Tạo một khuôn mẫu chung (lớp)

Lớp (Class)

- Tạo object bằng cách sử dụng special **function** kết hợp với từ khoá **this**

```
function Dog(dogName, weight, color, breed, type) {  
    this.dogName = dogName;  
    this.weight = weight;  
    this.color = color;  
    this.breed = breed;  
    this.type = type;  
    this.sua = function () {  
        return "Gau Gau";  
    };  
}  
  
let dog1 = new Dog("Jacky", 0.8, "brown", "chihuahua teacup", "short hair" );  
let dog2 = new Dog("Javascript", 1, "brown", "chihuahua mini", "long hair" );
```

Object và array

- Mảng các object (objects in arrays)

```
Dog[0] = new Dog("Jacky", 0.8, "brown", "chihuahua teacup", "short hair" );  
Dog[1] = new Dog("Javascript", 1, "brown", "chihuahua mini", "long hair" );  
Dog[2] = new Dog("Jscript", 1, "white", "chihuahua", "long hair" );  
Dog[3] = new Dog("Jscript", 1, "black & white", "chihuahua", "short hair" );
```

- Truy xuất đến thuộc tính và phương thức của đối tượng trong mảng đối tượng

```
Dog[1].sua()
```

```
Dog[1].dogName;
```

- Lặp qua các đối tượng (dùng for in)

```
for (var x in Dog) {  
    console.log(Dog[x].dogName);  
}
```

Object và array

- Mảng trong object (arrays in objects)

```
company = {  
    companyName: "Healthy Candy",  
    activities: ["food manufacturing",  
                "improving kids' health",  
                "manufacturing toys"],  
    yearOfEstablishment: 2021  
};
```

- Truy xuất

```
let activity = company.activities[1];
```

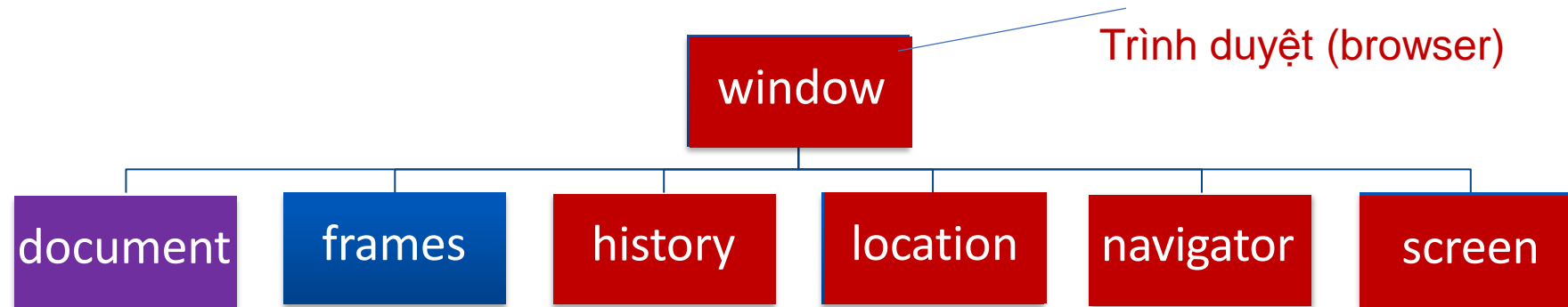


PHẦN II

BOM

Browser Object Model (BOM)

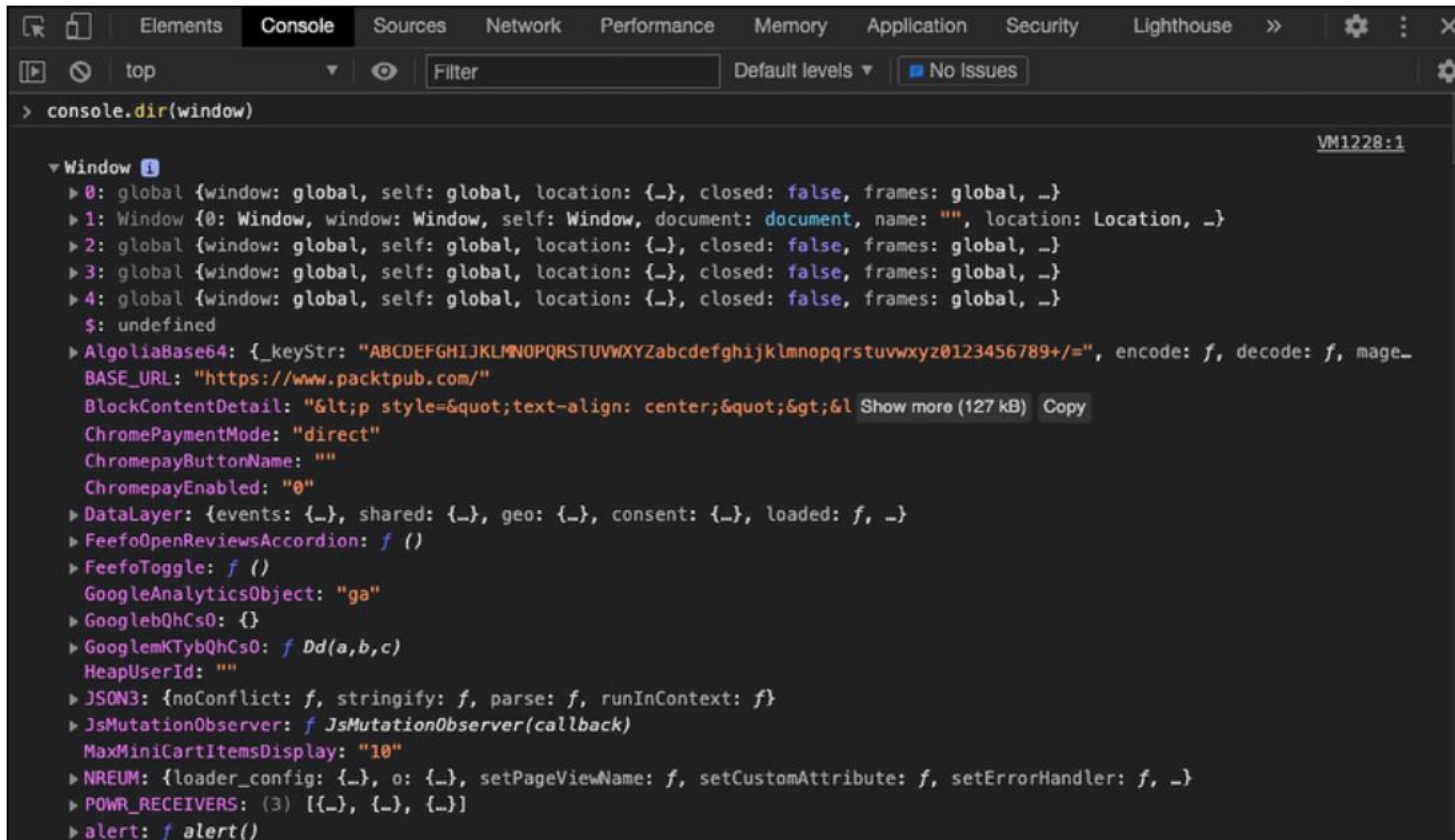
- BOM là một hệ thống phân cấp hình cây gồm các đối tượng



- Các đối tượng cung cấp thuộc tính và phương thức cho lập trình viên Javascript
- Đối với mỗi đối tượng, mỗi trình duyệt hỗ trợ các thuộc tính và phương thức khác nhau
 - Hiểu môi trường mà trình duyệt cung cấp để viết mã Javascript chạy ổn định trên nhiều trình duyệt

Browser Object Model (BOM)

- `Console.dir()`: hiển thị danh sách tất cả các thuộc tính của đối tượng đặc biệt



```
> console.dir(window)

▼ Window
  ▶ 0: global {window: global, self: global, location: {_, closed: false, frames: global, _}}
  ▶ 1: Window {0: Window, window: Window, self: Window, document: document, name: "", location: Location, _}
  ▶ 2: global {window: global, self: global, location: {_, closed: false, frames: global, _}}
  ▶ 3: global {window: global, self: global, location: {_, closed: false, frames: global, _}}
  ▶ 4: global {window: global, self: global, location: {_, closed: false, frames: global, _}}
  $: undefined
  ▶ AlgoliaBase64: {_keyStr: "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/", encode: f, decode: f, mage_
    BASE_URL: "https://www.packtpub.com/"
    BlockContentDetail: "<p style="text-align: center;">&l Show more (127 kB) Copy
    ChromePaymentMode: "direct"
    ChromepayButtonName: ""
    ChromepayEnabled: "0"
  ▶ DataLayer: {events: {_, shared: {_, geo: {_, consent: {_, loaded: f, _}}
  ▶ FeefoOpenReviewsAccordion: f ()
  ▶ FeefoToggle: f ()
    GoogleAnalyticsObject: "ga"
  ▶ GooglebQhCs0: {}
  ▶ GooglemKTybQhCs0: f Dd(a,b,c)
    HeapUserId: ""
  ▶ JSON3: {noConflict: f, stringify: f, parse: f, runInContext: f}
  ▶ JsMutationObserver: f JsMutationObserver(callback)
    MaxMiniCartItemDisplay: "10"
  ▶ NREUM: {loader_config: {_, o: {_, setPageViewName: f, setCustomAttribute: f, setErrorHandler: f, _}}
  ▶ POWR_RECEIVERS: (3) [{_, {_, {_}}
  ▶ alert: f alert()
```

Đối tượng window

- Window là đối tượng thể hiện cửa sổ hiển thị hiện tại của trình duyệt
- Một số phương thức của đối tượng window đã được sử dụng: `alert()`, `prompt()`, `confirm()`
- Các thuộc tính và phương thức của window có thể gọi trực tiếp hoặc thông qua window

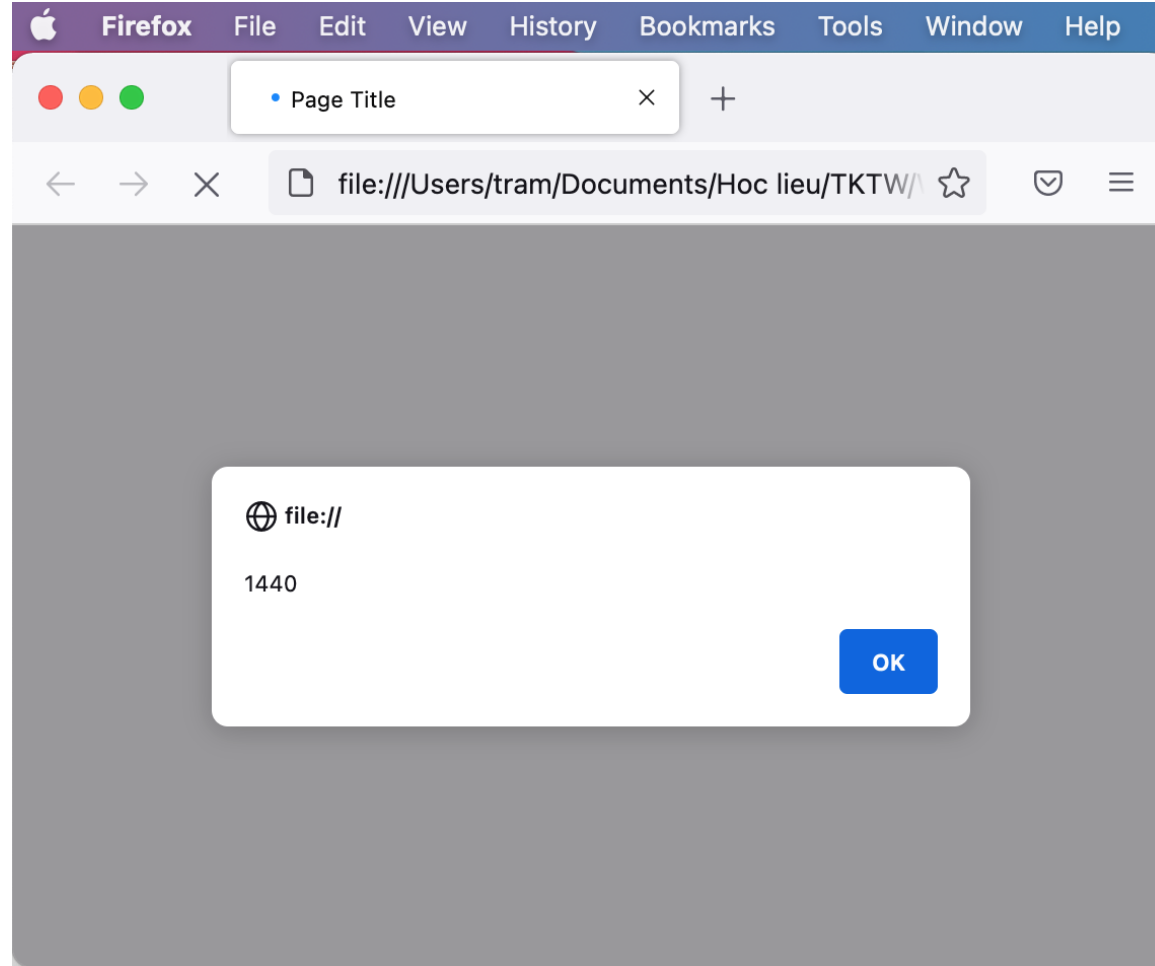
```
alert("Hello");
```

```
window.alert("Hello");
```

Các thuộc tính của Window

Thuộc tính	Giải thích
<code>closed</code>	Có giá trị là True khi cửa sổ được đóng
<code>defaultStatus</code>	Thiết lập văn bản mặc định trên thanh trạng thái của trình duyệt
<code>name</code>	Thiết lập hoặc trả về tên của cửa sổ
<code>opener</code>	Tham chiếu đến cửa sổ tạo ra cửa sổ hiện tại
<code>status</code>	Thông tin xuất hiện trên thanh trạng thái
<code>innerHeight</code>	Thiết lập hoặc trả về chiều cao phần nội dung của cửa sổ
<code>document</code>	Trả về đối tượng document của cửa sổ

Các thuộc tính của Window



```
alert(window.innerWidth);
```

Các phương thức của window

Phương thức	Giải thích
focus()	Chuyển focus đến cửa sổ
blur()	Bỏ focus đến cửa sổ
close()	Đóng cửa sổ
open()	Mở cửa sổ
print()	Thực hiện chức năng in
moveTo()	Sử dụng để chuyển cửa sổ về vị trí xác định
resizeTo()	Thay đổi kích thước cửa cửa sổ về vị trí xác định

Các phương thức của window

- Window.open: sử dụng để mở một cửa sổ từ cửa sổ hiện thời

`window.open(url, ten, dactinh)`

- url: url của trang web
- ten: tên của cửa sổ sẽ mở
- dactinh: các đặc tính mà cửa sổ được mở sẽ có (mỗi trình duyệt sẽ hỗ trợ một tập các đặc tính riêng)

```
window.open("http://www.google.com.vn/", "timkiem", "menubar =  
yes, width = 800, height = 600")
```

Chú ý:

- Chỉ nên sử dụng cách này khi thật cần thiết vì trình duyệt có thể bị disable javascript
- Có thể sử dụng thẻ <a> để thay thế

Window screen object (đối tượng screen)

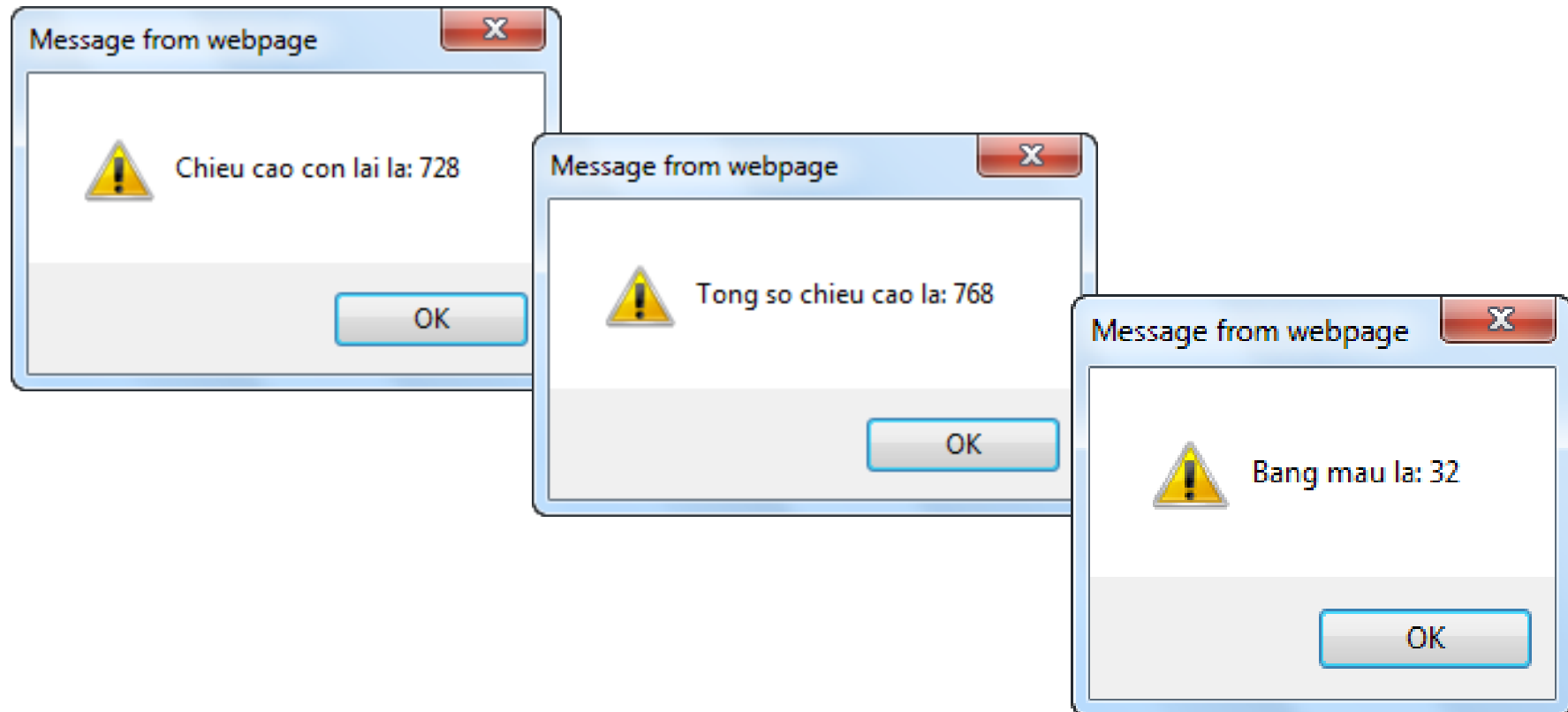
- Mỗi người truy cập sử dụng màn hình có độ phân giải khác nhau, kích thước khác nhau, dải màu khác nhau...
- ⇒ Người lập trình phải nắm được thông tin này để hiển thị ảnh phù hợp, hiển thị trang web có kích thước phù hợp...
- Đối tượng screen cung cấp thuộc tính để lấy thông tin về màn hình của người truy cập

Window screen object (đối tượng screen)

Thuộc tính	Giải thích
availHeight	Trả về chiều dài của màn hình (trừ kích thước của window taskbar)
availWidth	Trả về chiều rộng của màn hình (trừ kích thước của window taskbar)
height	Trả về chiều dài của màn hình
width	Trả về chiều rộng của màn hình
pixelDepth	Trả về độ phân giải của màn hình
colorDepth	Trả về bảng màu để hiển thị ảnh

DEMO

```
alert("Chieu cao con lai la: " + screen.availHeight);  
alert("Tong so chieu cao la: " + screen.height);  
alert("Bang mau la: " + screen.colorDepth);
```

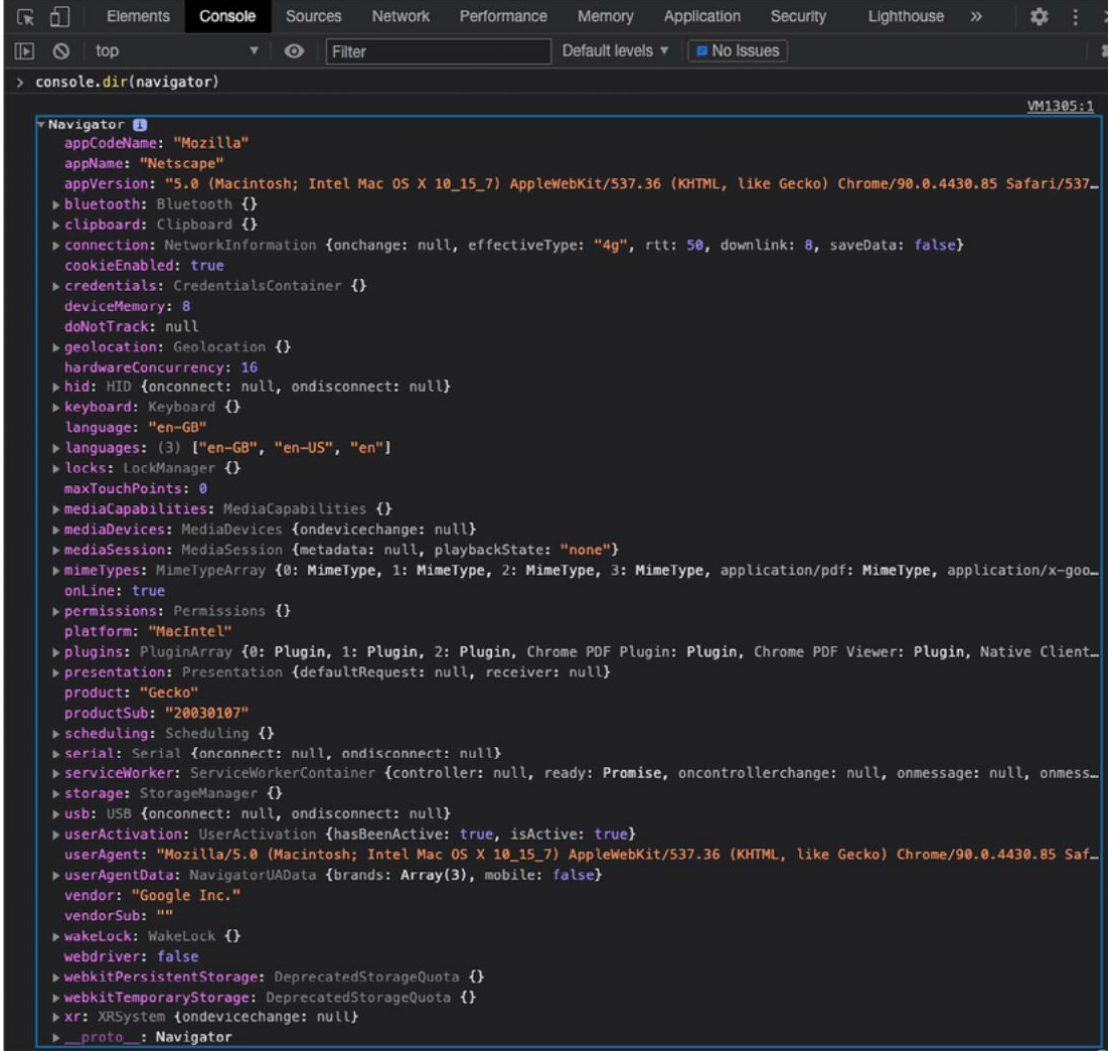


Window navigator object (đối tượng navigator)

- Mỗi trình duyệt có cách thức thi hành mã Javascript riêng
- Có thể cùng thực hiện một chức năng, nhưng đối với từng trình duyệt, cần phải viết các đoạn mã khác nhau.

⇒ Cần biết thông tin về trình duyệt để viết mã Javascript phù hợp

- Đối tượng Navigator cung cấp các thông tin về trình duyệt đang sử dụng (version, hệ điều hành mà trình duyệt đang chạy)



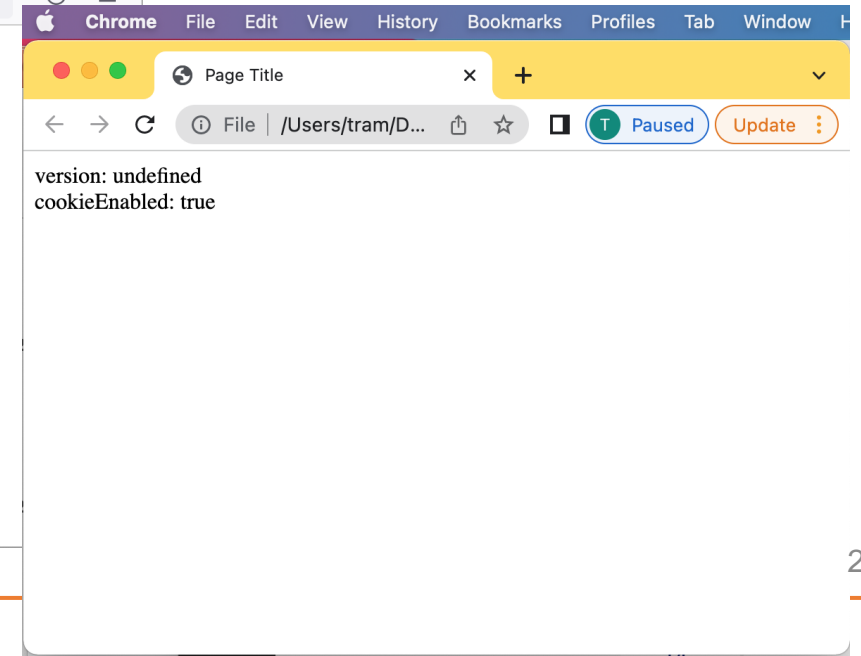
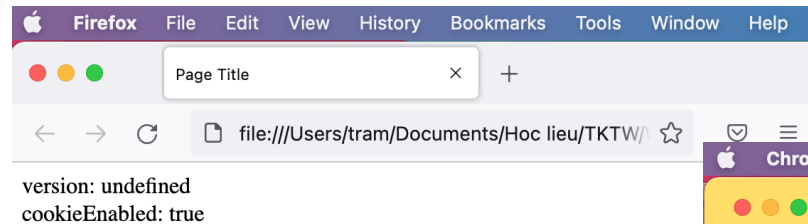
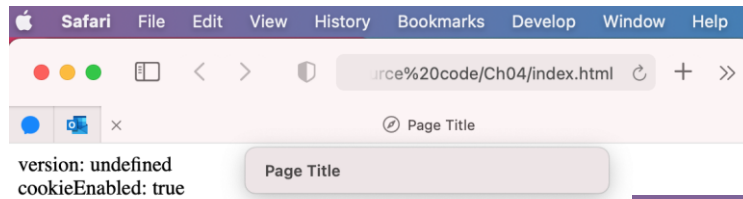
```
> console.dir(navigator)

VM1305:1
Navigator {
  appName: "Mozilla"
  appVersion: "Netscape"
  appVersion: "5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.85 Safari/537..."
  bluetooth: Bluetooth {}
  clipboard: Clipboard {}
  connection: NetworkInformation {onchange: null, effectiveType: "4g", rtt: 50, downlink: 8, saveData: false}
  cookieEnabled: true
  credentials: CredentialsContainer {}
  deviceMemory: 8
  doNotTrack: null
  geolocation: Geolocation {}
  hardwareConcurrency: 16
  hid: HID {onconnect: null, ondisconnect: null}
  keyboard: Keyboard {}
  language: "en-GB"
  languages: (3) ["en-GB", "en-US", "en"]
  locks: LockManager {}
  maxTouchPoints: 0
  mediaCapabilities: MediaCapabilities {}
  mediaDevices: MediaDevices {ondevicechange: null}
  mediaSession: MediaSession {metadata: null, playbackState: "none"}
  mimeTypes: MimeTypeArray {0: MimeType, 1: MimeType, 2: MimeType, 3: MimeType, application/pdf: MimeType, application/x-go...
  online: true
  permissions: Permissions {}
  platform: "MacIntel"
  plugins: PluginArray {0: Plugin, 1: Plugin, 2: Plugin, Chrome PDF Plugin: Plugin, Chrome PDF Viewer: Plugin, Native Client...
  presentation: Presentation {defaultRequest: null, receiver: null}
  product: "Gecko"
  productSub: "20030107"
  scheduling: Scheduling {}
  serial: Serial {onconnect: null, ondisconnect: null}
  serviceWorker: ServiceWorkerContainer {controller: null, ready: Promise, oncontrollerchange: null, onmessage: null, onmess...
  storage: StorageManager {}
  usb: USB {onconnect: null, ondisconnect: null}
  userActivation: UserActivation {hasBeenActive: true, isActive: true}
  userAgent: "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.85 Saf...
  userAgentData: NavigatorUAData {brands: Array(3), mobile: false}
  vendor: "Google Inc."
  vendorSub: ""
  wakeLock: WakeLock {}
  webdriver: false
  webkitPersistentStorage: DeprecatedStorageQuota {}
  webkitTemporaryStorage: DeprecatedStorageQuota {}
  xr: XRSystem {ondevicechange: null}
  __proto__: Navigator
}
```

`console.dir(window.navigator);` **HOẶC** `console.dir(navigator);`

DEMO

```
document.write("version: " + navigator.version + "<br>");  
document.write("cookieEnabled: " + navigator.cookieEnabled);
```

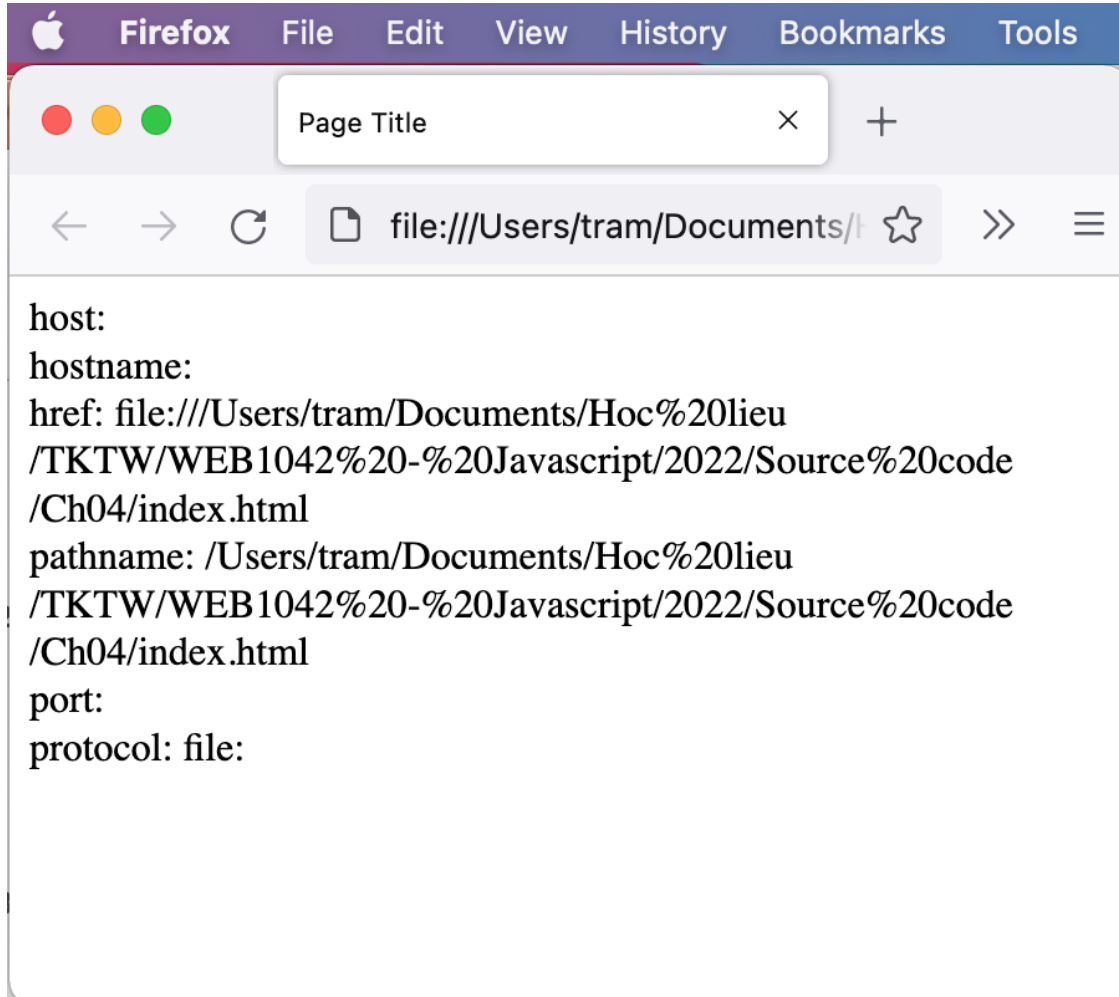


Window location object (Đối tượng location)

- Quản lý thông tin về URL hiện tại

Thuộc tính	Giải thích
host	Trả về tên host và cổng của URL
hostname	Trả về tên host
href	Trả về toàn bộ URL
pathname	Trả về tên đường dẫn của URL
port	Trả về cổng mà server sử dụng cho URL
protocol	Trả về protocol của URL
Phương thức	Giải thích
assign()	Load document mới
reload()	Load lại document hiện tại

DEMO



```
document.write("host: " + location.host +  
"<br>");  
document.write("hostname: " +  
location.hostname + "<br>");  
document.write("href: " + location.href +  
"<br>");  
document.write("pathname: " +  
location.pathname + "<br>");  
document.write("port: " + location.port +  
"<br>");  
document.write("protocol: " +  
location.protocol + "<br>");
```

Window history object (Đối tượng history)

- Chứa thông tin về các URL được người dùng truy cập

Thuộc tính	Giải thích
length	Trả về số lượng URL trong danh sách History

Phương thức	Giải thích
back()	Load URL trước đó trong danh sách History
forward()	Load URL sau đó trong danh sách History
go()	Load URL cụ thể từ History

DEMO

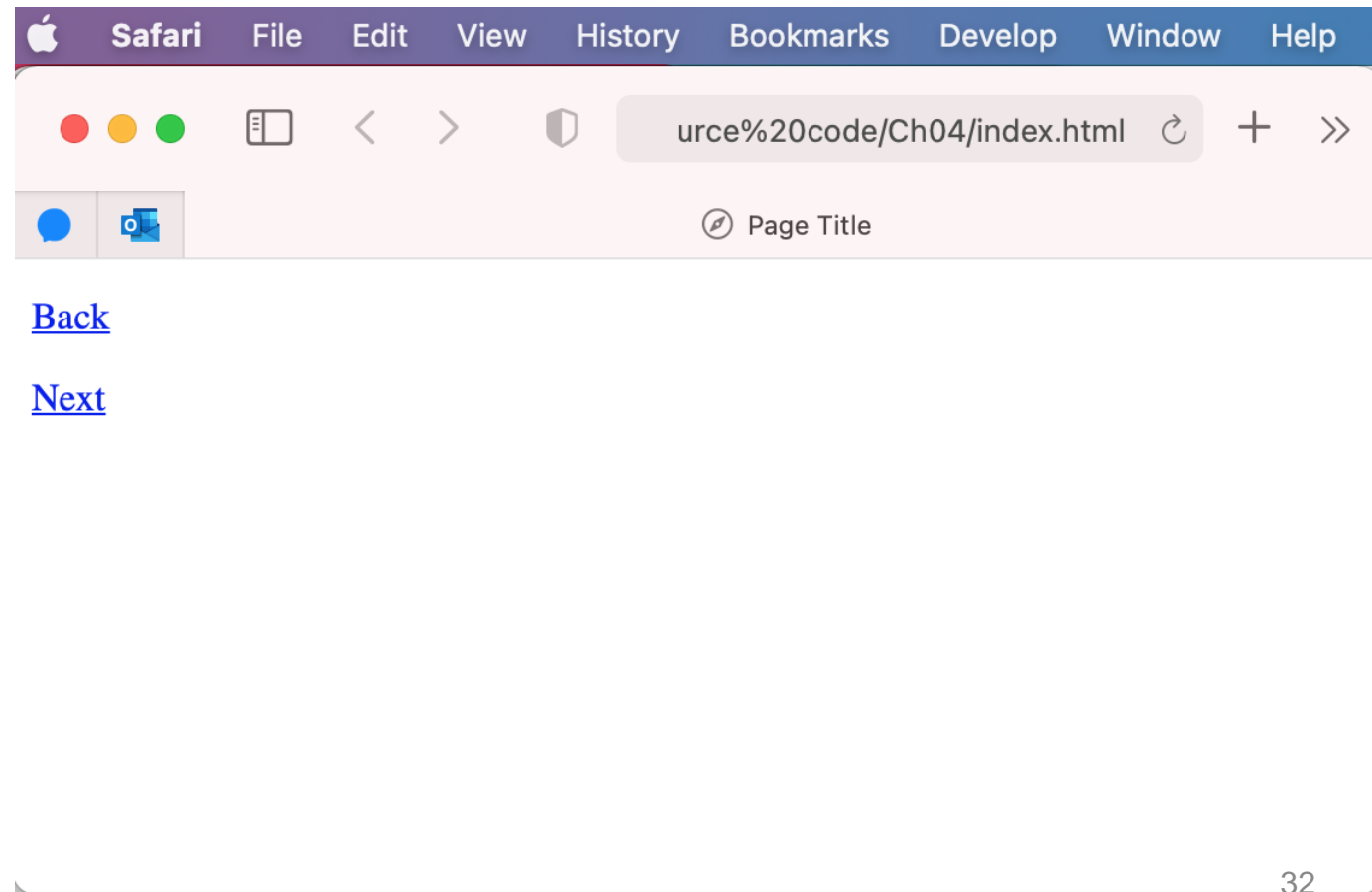
- Định nghĩa hàm trong thẻ Javascript

```
function goBack() {  
    history.back();  
}
```

```
function goNext() {  
    history.forward();  
}
```

- Gọi hàm

```
<p><a href = "#"   
onclick="goBack()">Back</a></p>  
<p><a href = "#"   
onclick="goNext()">Next</a></p>
```



TIMER

- Javascript cung cấp các phương thức để xử lý sự kiện thời gian
- Các phương thức này thuộc đối tượng window
- Một số phương thức quan trọng

Phương thức	Giải thích
setTimeout	Thực hiện công việc sau một khoảng thời gian trong tương lai
clearTimeout	Hủy bỏ setTimeout trước đó
setInterval	Thực hiện lặp lại công việc sau một khoảng thời gian
clearInterval	Hủy bỏ setInterval

TIMER

- Cú pháp

```
var t=setTimeout("Lệnh_javascript",số_mili_giây );
```

- Lệnh_javascript: mã thực thi hoặc lời gọi hàm
- Mili_giây: sau thời gian này mã sẽ được thực hiện
- setTimeout() trả về giá trị, giá trị được lưu trong biến t. Nếu muốn huỷ bỏ setTimeout, sử dụng hàm clearTimeout và truyền vào đối số t
- Cú pháp tương tự đối với setInterval()

DEMO setTimeout

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Page Title</title>
  <script type="text/javascript">
    function onClickEvent() {
      var t = setTimeout("alert('Hi');", 1000);
    }
  </script>
</head>
<body>
  <button onclick="onClickEvent();"> Click here!</button>
</body>
</html>
```

DEMO setInterval

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Page Title</title>
  <script type="text/javascript">
    function onClickEvent() {
      var t = setInterval("alert('Hi');", 1000);
    }
  </script>
</head>
<body>
  <button onclick="onClickEvent();"> Click here!</button>
</body>
</html>
```

TỔNG KẾT

- Có rất nhiều phương thức lập trình. Mỗi phương thức phù hợp cho một mục đích riêng. Phương thức lập trình hướng đối tượng được phát triển rộng rãi nhất.
- Mỗi đối tượng có các thuộc tính và phương thức riêng
- Các đối tượng có các thuộc tính và phương thức giống nhau thuộc cùng một lớp
- Browser Object Model (BOM) là tập hợp các đối tượng được xây dựng sẵn giúp lập trình viên thao tác với trình duyệt

TỔNG KẾT

- Trình duyệt được biểu diễn bằng đối tượng window
- Đối tượng window có các đối tượng con là document, frames, history, location, navigator, screen
- Đối tượng document đại diện cho nội dung trang web
- Đối tượng history chứa thông tin về các url được người dùng truy cập
- Đối tượng location chứa thông tin về url hiện tại
- Đối tượng navigator chứa thông tin về trình duyệt
- Đối tượng screen chứa thông tin về màn hình

*Thank
you!*