



JAVASCRIPT

TH.S LÊ QUANG SONG

Request

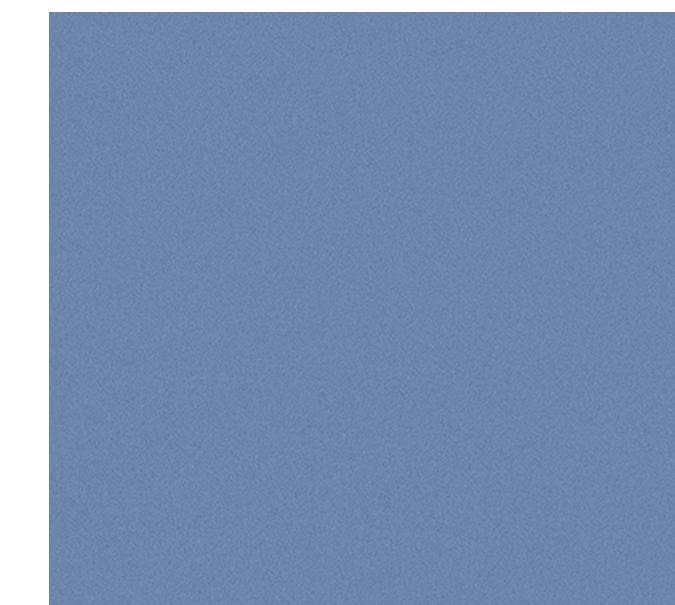
Client

- Script*
 - * *vbscript*
 - Javascript*
- * **Hình ảnh**
- * **Javascript**
- * **CSS (abc.css)**
- * **Font**

bootstrap.min.js
bootstrap.js

Server

- Code server*
 - * *php*
 - * *asp.net*
 - * *nodejs*



Response

Script là gì ?

● Client-Side Script

- Script được thực thi tại *Client-Side* (trình duyệt):
Thực hiện các tương tác với người dùng (tạo menu chuyển động, ...), kiểm tra dữ liệu nhập,...

● Server-Side Script

- Script được xử lý tại *Server-Side*, nhằm tạo các trang web có khả năng phát sinh nội dung động.
Một số xử lý chính: kết nối CSDL, truy cập hệ thống file trên server, phát sinh nội dung html trả về người dùng...

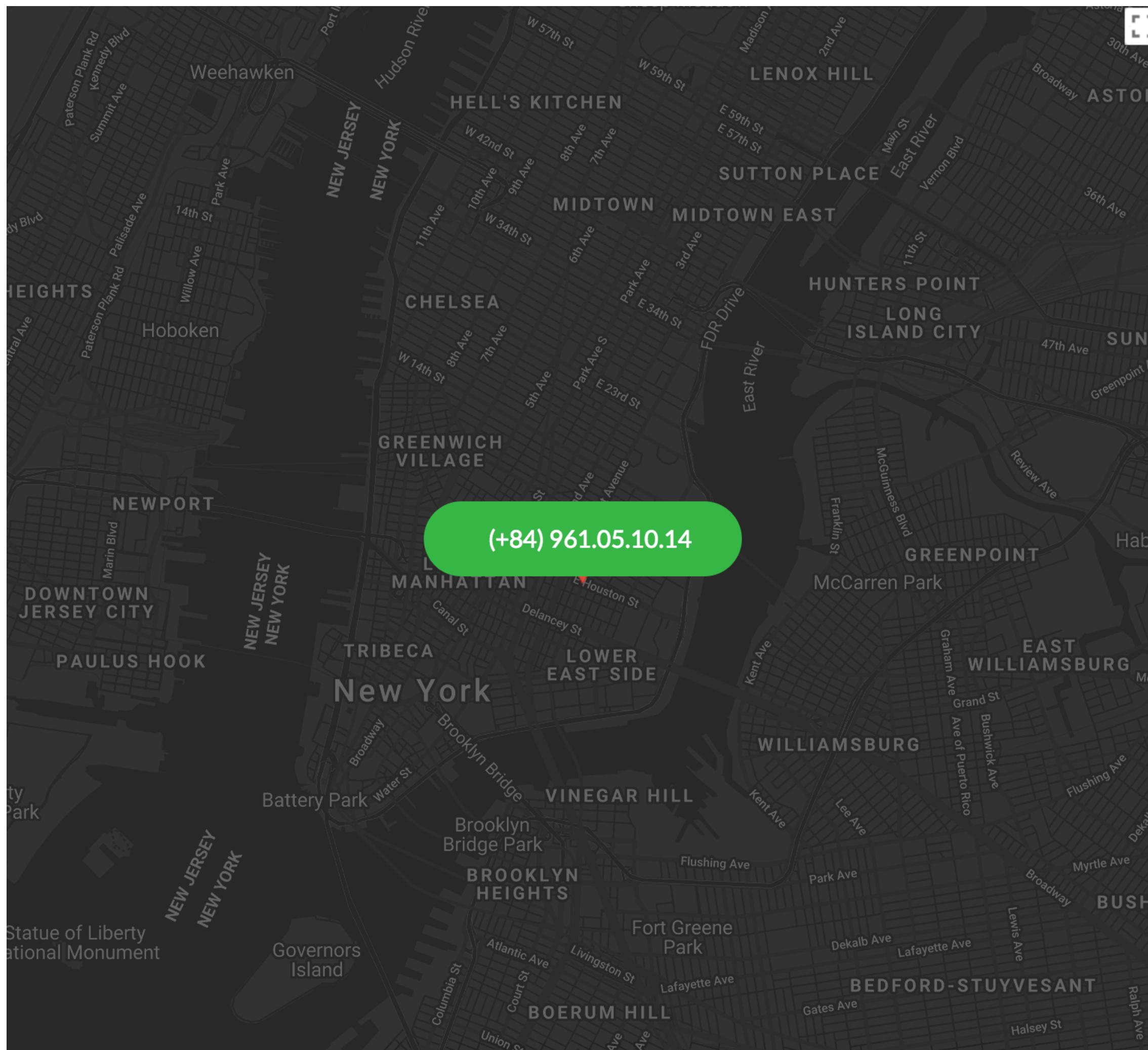


JavaScript là gì ?

- **JavaScript** Là ngôn ngữ *Client-side script* hoạt động trên trình duyệt của người dùng (*client*)
- **Chia sẻ xử lý** trong ứng dụng web. Giảm các xử lý không cần thiết trên server.
- Giúp **tạo các hiệu ứng, tương tác cho** trang web.



DỰ ÁN 1 - TÍNH TIỀN TAXI



TÍNH CƯỚC UBER

Vui lòng chọn loại xe:



Uber X



Uber SUV



Uber Black

Nhập vào số KM

Số KM ...

Nhập thời gian chờ

TÍNH TIỀN

IN HÓA ĐƠN

5

DỰ ÁN 1 - TÍNH TIỀN TAXI

MÔ TẢ DỰ ÁN

- * Chọn loại xe
- * Nhập vào số KM đi được
- * Nhấn Tính tiền → Tổng tiền phải trả
- * Nhấn In Hóa Đơn → In chi tiết tiền theo số KM nhập vào

GIÁ CƯỚC ÁP DỤNG

KHOẢNG CÁCH	LOẠI UBER		
	UBER X	UBER SUV	UBER BLACK
Mở cửa/1 km đầu tiên	8.000 vnđ	9.000 vnđ	10.000 vnđ
Từ 1km đến 20km	12.000 vnđ	14.000 vnđ	16.000 vnđ
Từ 21 trở đi	10.000 vnđ	12.000 vnđ	14.000 vnđ
Thời gian chờ	2.000 vnđ/1 phút	3.000 vnđ/1 phút	4.000 vnđ/1 phút

6

DỰ ÁN 1 - TÍNH TIỀN TAXI

MÔ TẢ DỰ ÁN

- * Nhấn Tính tiền → Tổng tiền phải trả và xuất vào nội dung thẻ p chứa trong div bên dưới
- * Nhấn In Hóa Đơn → In chi tiết tiền theo số KM nhập vào

CHI TIẾT HÓA ĐƠN

CHI TIẾT	SỬ DỤNG	ĐƠN GIÁ	THÀNH TIỀN
TỔNG CỘNG: vnd			

7

DỰ ÁN 1 - TÍNH TIỀN TAXI

KIẾN THỨC SAU DỰ ÁN:

- * Tổ chức dự án có sử dụng Javascript
- * Biến và giá trị trong Javascript
- * Hàm trong Javascript
- * Gõ rối trong Javascript
- * Lệnh console.log
- * Lệnh alert()
- * Sự kiện trong nút
- * Biểu thức trong Javascript
- * Các lệnh điều kiện if...else
- * Các lệnh switch....case
- * DOM trong javascript
- * document.getElementById
- * Truy xuất & thay đổi thuộc tính style của thẻ thông qua DOM

8

DỰ ÁN 1 - TÍNH TIỀN TAXI

KIẾN THỨC SAU DỰ ÁN:

- *Cách để có nhanh giao diện
- *Khởi tạo dự án
- *Tạo thói quen chuẩn hóa khi code
- *Hiệu chỉnh và gắn layout
- *Phân tích bài toán (đầu vào, ra, giải thuật)

DEMO TRỰC TIẾP DỰ ÁN

REVIEW KIẾN THỨC

Nhúng JavaScript vào trang HTML



Định nghĩa Script trực tiếp trong trang HTML:

```
<script type="text/javascript">
```

```
<!--
```

// Lệnh Javascript

```
-->
```

```
</script>
```

Nhúng sử dụng script cài đặt từ 1 file **.js** khác:

```
<script src="abc.js"></script>
```

12



Nhúng JavaScript vào trang HTML



- Web Browser sẽ thực thi **<script>** khi load trang web theo thứ tự từ *trên xuống* dưới.
- Đối với *Source code JavaScript* có thể đặt trong các file *.js* sẽ được *nhúng* vào file HTML trước khi hoạt động.
- Các đoạn *code JavaScript* được Browser xử *cùng thứ tự* với các thẻ *HTML*. Trừ các **hàm** (*function*) chỉ được thực thi khi *có lời gọi hàm*.



Nhúng JavaScript vào trang HTML

```
<html>
  <head>
    <script type="text/javascript">
      some statements
    </script>
  </head>
  <body>
    <script type="text/javascript">
      some statements
    </script>
    <script src="Tên_file_script.js"> |</script>
    <script type="text/javascript">
      // gọi thực hiện các phương thức được định nghĩa
      // trong "Tên_file_script.js"
    </script>
  </body>
</html>
```

14



Nhúng JavaScript vào trang HTML



- Đặt giữa tag `<head>` và `</head>`: script sẽ thực thi ngay khi trang web được mở.
- Đặt giữa tag `<body>` và `</body>`: script trong phần body được thực thi khi trang web đang mở (sau khi thực thi các đoạn script có trong phần `<head>`).
- Số lượng đoạn client-script chèn vào trang không hạn chế.

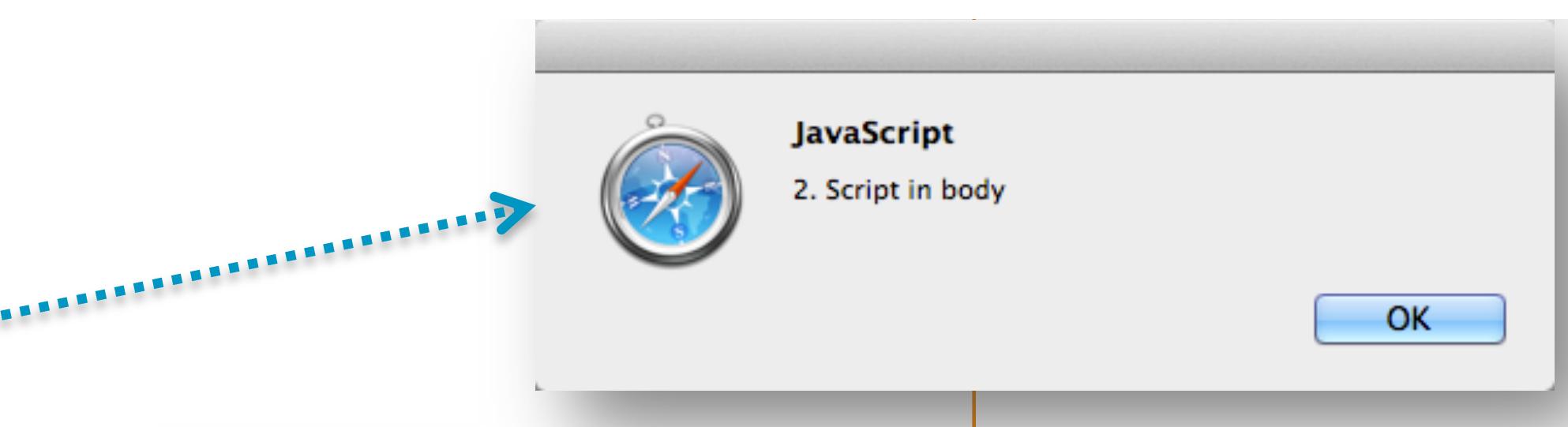
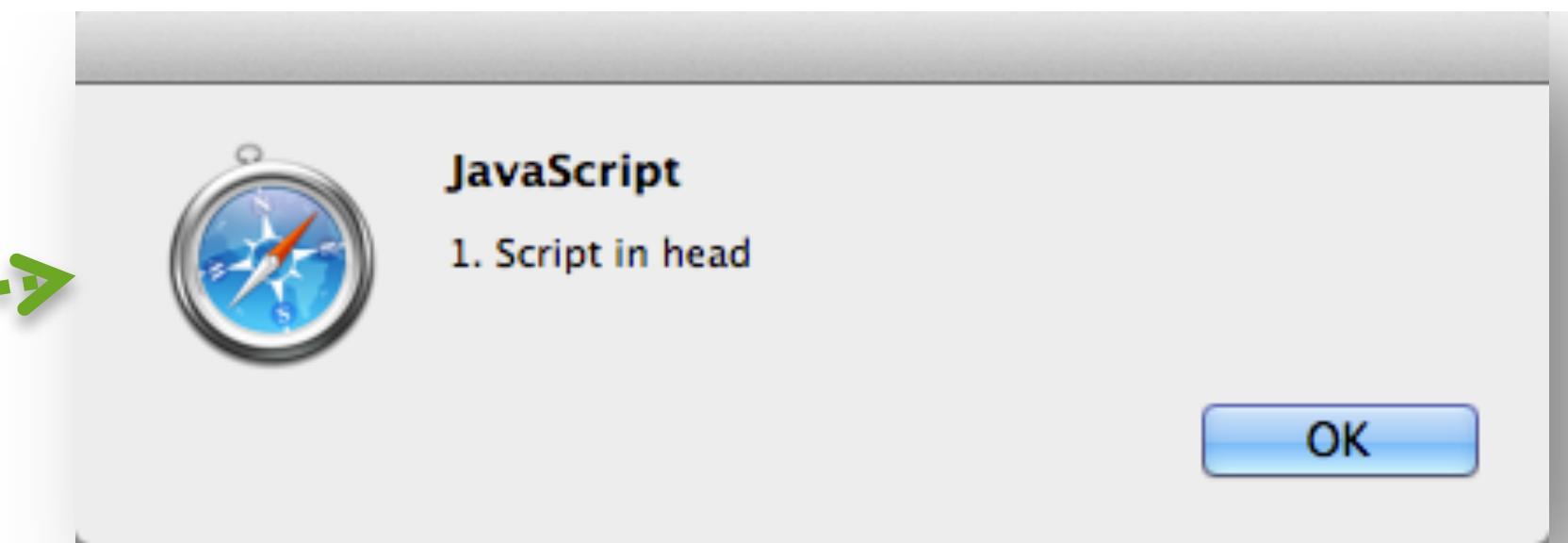


Nhúng JavaScript vào trang HTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8" />
    <title>Title of website</title>
    <script type="text/javascript">
      alert("1. Script in head");
    </script>
  </head>
```

```
<body>
  <script type="text/javascript">
    alert("2. Script in body");
  </script>
```

```
<script type="text/javascript" src="myScript.js" ></script>
<script type="text/javascript">
  showMessage();
</script>
</body>
</html>
```



Đọc lập trình t



KIỂM TRA DỮ LIỆU VÀ CÚ PHÁP

NGÔN NGỮ JAVASCRIPT



Biến số trong Javascript



● Cách đặt tên biến

- Bắt đầu bằng một chữ cái hoặc dấu _
- A..Z,a..z,0..9,_ : phân biệt HOA, Thường

Ví dụ: **var chuoisо;**

var ChuoiSo;

● Khai báo biến

- Sử dụng từ khóa **var**

Ví dụ: **var count = 10, amount;**

- *Không cần khai báo biến trước khi sử dụng*, biến thật sự tồn tại khi bắt đầu sử dụng lần đầu tiên.

- Biến *không cần khai báo kiểu dữ liệu* vì biến trong javascript *không có kiểu dữ liệu nhất định*



Kiểu dữ liệu trong Javascript

Kiểu dữ liệu	Ví dụ	Mô tả
Object	var listBooks = new Array(10) ;	Trước khi sử dụng, phải cấp phát bằng từ khóa new
String	“The cow jumped over the moon.” “40”	Chứa được chuỗi unicode Chuỗi rỗng “”
Number	0.066218 12	Theo chuẩn IEEE 754
boolean	true / false	
undefined	var myVariable ;	myVariable = undefined
null	connection.Close();	connection = null



Đổi kiểu dữ liệu



- Biến tự đổi kiểu dữ liệu khi giá trị mà nó lưu trữ thay đổi

Ví dụ:

```
var x = 10;           // x kiểu Number  
x = "hello world !"; // x kiểu String
```

- Có thể cộng 2 biến khác kiểu dữ liệu

Ví dụ:

```
var x;  
x = "12" + 34.5; // KQ: x = "1234.5"
```

- Hàm **parselnt(...)**, **parseFloat(...)** : Đổi KDL từ chuỗi sang số.



Hàm trong javascript

- Dạng thức khai báo chung:

```
function Tên_hàm(thamso1, thamso2, ...)  
{  
    ...  
}
```

- Hàm có giá trị trả về:

```
function Tên_hàm(thamso1, thamso2, ...)  
{  
    ...  
    return (value);  
}
```



Ví dụ Hàm trong javascript

- Ví dụ:

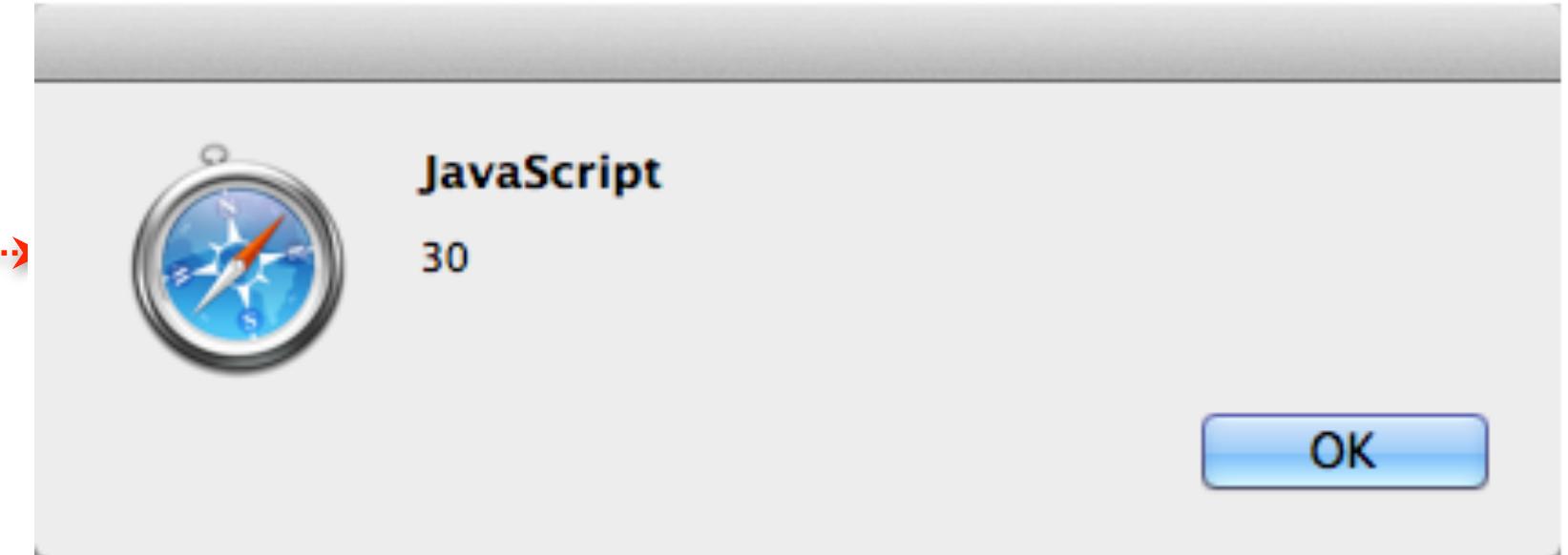
```
function Sum(x, y)
```

```
{
```

```
    tong = x + y;
```

```
    return tong;
```

```
}
```



- Gọi hàm:

```
var x = Sum(10, 20);
```

```
    alert(x);
```



Các qui tắc chung

- Khối lệnh được bao trong dấu {}
- Mỗi lệnh nên kết thúc bằng dấu ;
- Cách ghi chú thích:
 - // Chú thích 1 dòng
 - /* Chú thích
nhiều dòng */

Một số sự kiện thông dụng

- Các sự kiện được hỗ trợ bởi hầu hết các đối tượng
 - **onClick**
 - **onFocus**
 - **onChange**
 - **onBlur**
 - **onMouseOver**
 - **onMouseOut**
 - **onMouseDown**
 - **onMouseUp**
 - **onLoad**
 - **onSubmit**



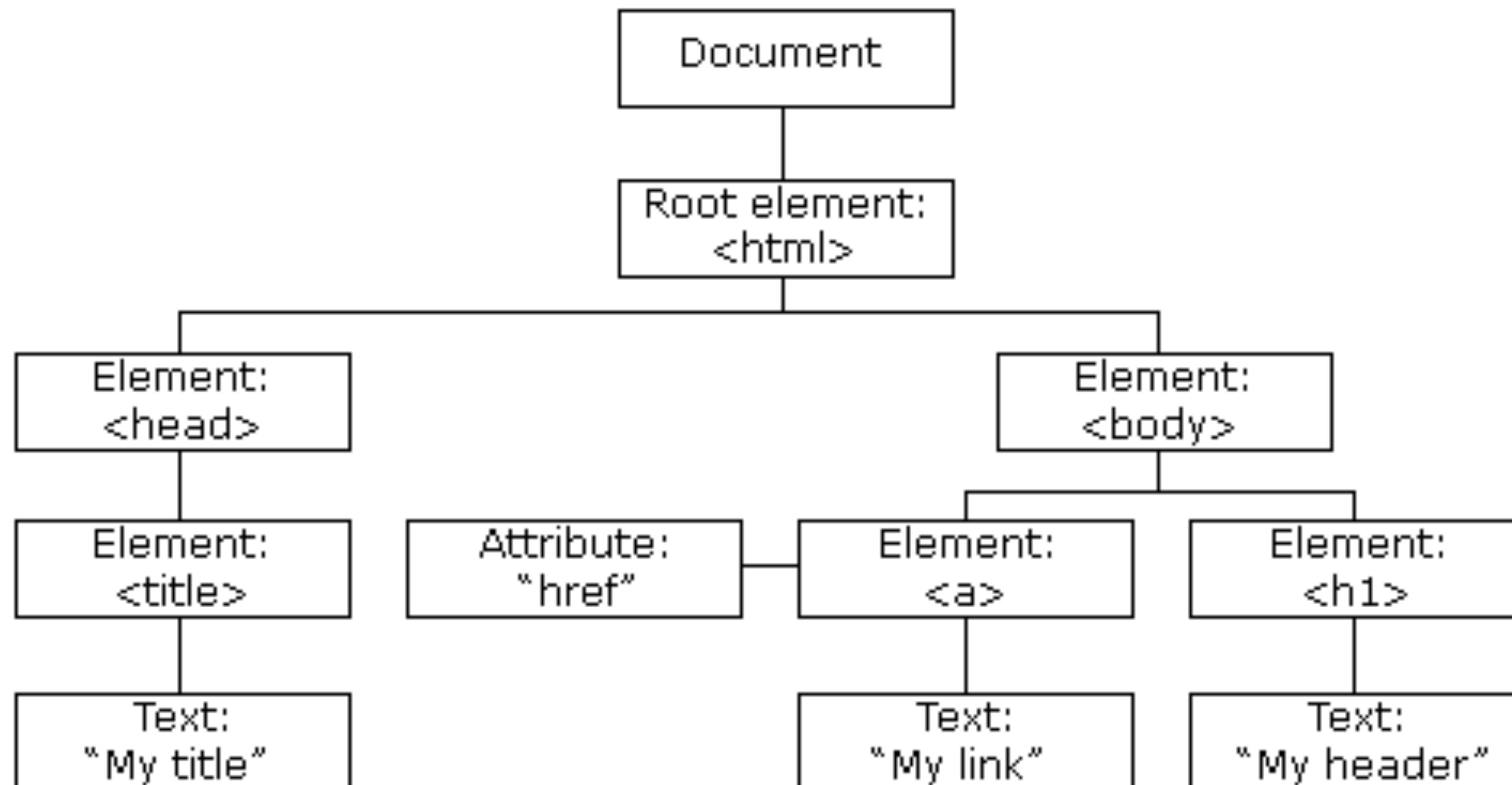
Đối tượng HTML DOM



- **DOM** = Document Object Model
- Là tập hợp các đối tượng HTML chuẩn được dùng để **truy xuất** và **thay đổi thành phần** HTML trong trang web (thay đổi nội dung tài liệu của trang)
- Một số đối tượng của DOM: **window, document, history, link, form, frame, location, event, ...**



ĐỐI TƯỢNG DOM



Đối tượng Window - DOM



- Là thể hiện của đối tượng **cửa sổ trình duyệt**
- Tồn tại khi mở 1 tài liệu HTML
- Sử dụng để truy cập thông tin của các đối tượng trên cửa sổ trình duyệt (tên trình duyệt, phiên bản trình duyệt, thanh tiêu đề, thanh trạng thái ...)



ĐỐI TƯỢNG DOM



Đối tượng Window - DOM



Properties

- **document**
- event
- history
- location
- name
- navigator
- screen
- status

Methods

- Alert
- Confirm
- Prompt
- Blur
- close
- Focus
- open



Đối tượng Document - DOM



- Biểu diễn cho **nội dung trang HTML** đang được hiển thị trên trình duyệt
- Dùng để lấy thông tin về tài liệu, các thành phần HTML và xử lý sự kiện



Đối tượng Document - DOM



- **Properties**
 - aLinkColor
 - bgColor
 - **body**
 - fgColor
 - linkColor
 - title
 - **URL**
 - vlinkColor
 - forms[]
 - images[]
 - **childNodes[]**
- **Methods**
 - **documentElement**
 - cookie
 -
 - close
 - open
 - **createTextNode(" text ")**
 - **createElement("HTMLtag")**
 - **getElementById("id")**
 - ...



TÌM PHẦN TỬ/THẺ (ELEMENT) HTML



PHƯƠNG THỨC	MÔ TẢ
<code>document.getElementById(id)</code>	Tìm thẻ thông qua Id của thẻ
<code>document.getElementsByTagName(name)</code>	Tìm TẤT CẢ các thẻ thông qua tên thẻ
<code>document.getElementsByClassName(name)</code>	Tìm TẤT CẢ các thẻ thông qua CSS class

3 4 5



THAY ĐỔI NỘI DUNG HTML CỦA THẺ



PHƯƠNG THỨC

MÔ TẢ

`element.innerHTML = new html content`

Thay đổi nội dung HTML bên trong của thẻ

`element.style.property = new style`

Thay đổi Style cho thẻ



THÊM VÀ XOÁ THẺ



PHƯƠNG THỨC	MÔ TẢ
<code>document.createElement(element)</code>	Tạo một thẻ mới
<code>element.removeChild(element)</code>	Xoá một thẻ
<code>document.appendChild(element)</code>	Thêm một thẻ
<code>document.replaceChild(element)</code>	Thay thế một thẻ
<code>document.write(text)</code>	Viết ngay vào HTML

DOM & CSS

Thiết lập CSS bằng JavaScript

Cú pháp:

```
document.getElementById("object").style.[tên thuộc tính] = '[giá trị]';
```

Lấy giá trị CSS bằng JavaScript

```
var value = document.getElementById("object").style.[tên thuộc tính];
```

Nếu thuộc tính có dấu gạch ngang như: *font-size, line-height, margin-bottom* thì thuộc tính đó trong *style* sẽ có tên là *fontSize, lineHeight, marginBottom*



DỰ ÁN 1 - TÍNH TIỀN UBER - PHẦN 2

TÍNH VÀ XUẤT HÓA ĐƠN

KIẾN THỨC SAU DEMO

- * Cách gắn table bootstrap có hover
- * Tạo động thẻ sử dụng **createElement, appendChild**
- * **Phương pháp tối ưu code**

CYBERSOFT
Admin

- Dashboard
- Components (12)
- UI Elements
- Widgets 08
- Mailbox
- Tables 05
- Forms checked
- Pages 02
- Charts new

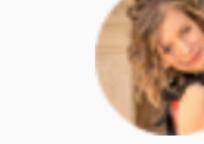
Search...

Today Sales 45

Today Visitors 80

Today Orders 51

QUẢN LÝ SINH VIÊN

<input type="checkbox"/>	HÌNH ẢNH	TÊN	ĐIỂM TRUNG BÌNH	THAO TÁC
<input type="checkbox"/>		Nguyễn Văn A	9.4	 
<input type="checkbox"/>		Đỗ Văn Nhơn	8.2	 
<input type="checkbox"/>		Vin Thị Liên	3.4	 
<input type="checkbox"/>		Đen Thị Giàu	6.4	 

DỰ ÁN 2 - QUẢN LÝ SINH VIÊN

YÊU CẦU DỰ ÁN

- * *Đọc thông tin ĐTB từ table vào mảng*
- * *Sử dụng thuật toán tìm kiếm để tìm SV có điểm trung bình lớn nhất*
- * *In ra Tên Sinh viên + Hình Sinh viên + ĐTB của Sinh viên có điểm lớn nhất*
- * *Sử dụng thuật toán Đếm để đếm có bao nhiêu SV yếu.*
- * *Liệt kê ra danh sách sinh viên yếu bên dưới.*

DỰ ÁN 2 - QUẢN LÝ SINH VIÊN

KIẾN THỨC SAU DỰ ÁN:

- * Sử dụng tài nguyên **w3layout** hoặc từ khóa tìm kiếm
- * Vòng lặp **for, while**
- * Thuộc tính **rows, cells, lặp trong table**
- * Mảng trong Javascript
- * Thuật toán tìm kiếm tuyến tính
- * Tìm **Max** trong mảng
- * Giải thuật **Đếm** trong mảng

DỰ ÁN 3 - QUẢN LÝ NHÂN VIÊN - HỌC VIÊN LÀM

<input type="checkbox"/>	HÌNH ẢNH	TÊN	CHỨC VỤ	SỐ NGÀY LÀM	LƯƠNG/NGÀY	PHỤ CẤP	THAO TÁC
<input type="checkbox"/>		Nguyễn Văn A	Sếp	22	200	500	 
<input type="checkbox"/>		Đỗ Văn Nhơn	Trưởng phòng	25	180	400	 
<input type="checkbox"/>		Vin Thị Liên	Nhân viên	28	150	300	 
<input type="checkbox"/>		Đen Thị Giàu	Nhân viên	24	150	300	 
<input type="checkbox"/>		Ngô Văn Mạnh	Nhân viên	30	150	300	 
<input type="checkbox"/>		Văn Sỹ Lâm	Trưởng phòng	28	150	400	 
<input type="checkbox"/>		Bùi Đình Sen	Nhân viên	18	150	300	 

TỰ LÀM

DỰ ÁN 3 - QUẢN LÝ NHÂN VIÊN - HỌC VIÊN LÀM

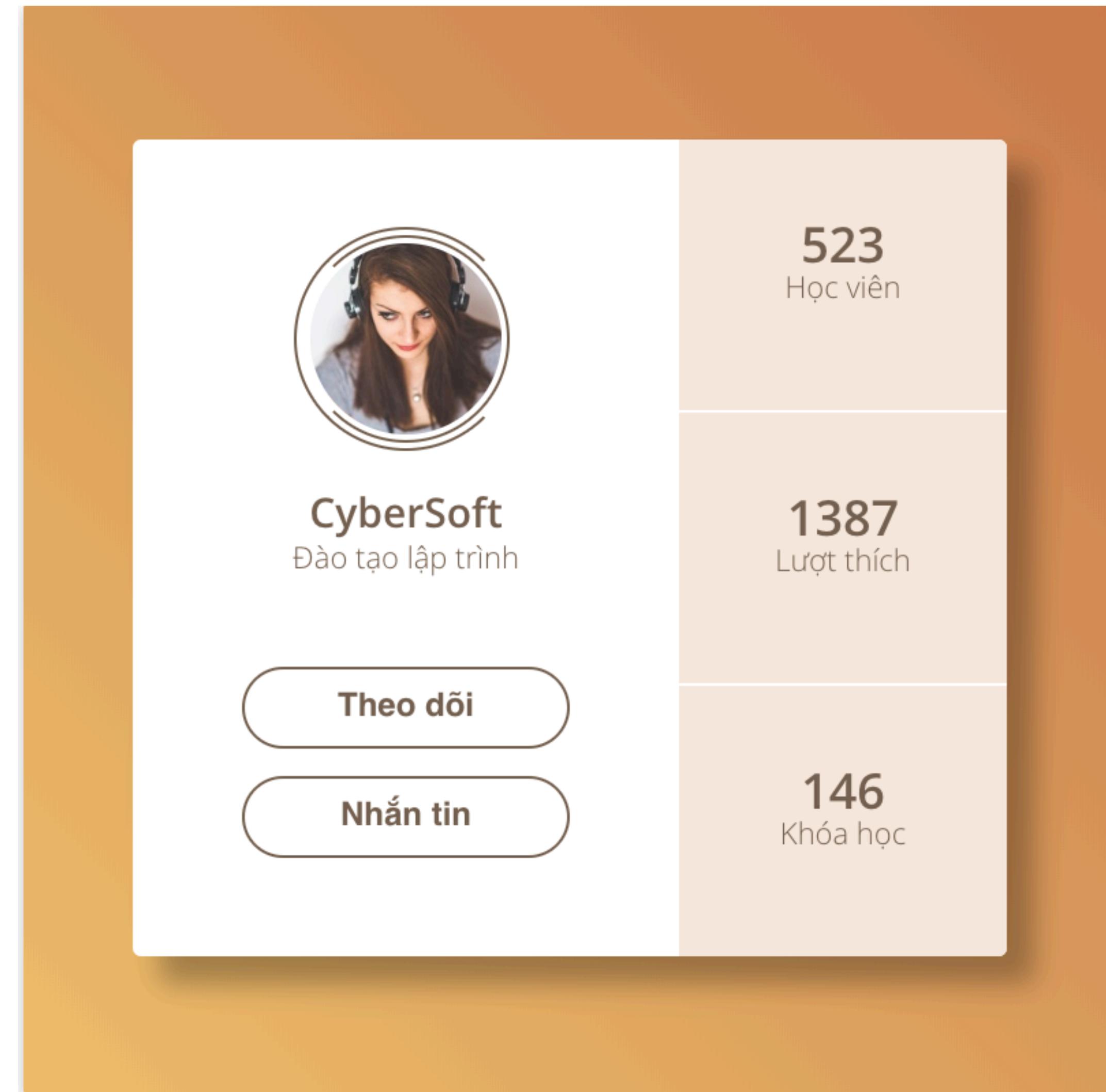
YÊU CẦU DỰ ÁN

- * Tạo mảng dữ liệu chứa các dòng dữ liệu trong table đã cho
- * Lương được tính theo công thức:

Số ngày làm * Lương/ngày + phụ cấp

- * Xuất thông tin **Nhân viên** (Hình + Tên + Lương tháng) có lương cao nhất trong vùng div bên dưới
- * Xuất thông tin **Trưởng phòng** (Hình + Tên + Lương tháng) có lương cao nhất trong vùng div bên dưới
- * Tính tổng tiền lương phải trả cho tất cả nhân viên (nhân viên, sếp, trưởng phòng).

JAVASCRIPT OOP



JAVASCRIPT OOP-SIMPLE OBJECT

Yêu cầu dự án

- * Xây dựng đối tượng để quản lý thông tin
- * Lấy dữ liệu từ mảng cho trước -> đổ thông tin vào đối tượng

DỰ ÁN 2 - QUẢN LÝ SINH VIÊN

KIẾN THỨC SAU DỰ ÁN:

- * Hiểu về *simple object* trong Javascript
- * Sử dụng sự kiện động cho thẻ - ***addEventListener***

REVIEW - SIMPLE OBJECT TRONG JAVASCRIPT

KIẾN THỨC SAU DỰ ÁN:

- * Các kiểu dữ liệu cơ bản : String, number, boolean, null, undefined
-> **Không mô tả được các đối tượng thực tế**

Các kiểu dữ liệu cơ bản : String, number, boolean, null, undefined

Object

Properties

Methods



car.name = Fiat

car.start()

car.model = 500

car.drive()

car.weight = 850kg

car.brake()

car.color = white

car.stop()

- Các biến trong Object gọi là : **Property ~ Thuộc tính**
- Các hàm trong Object là : **Method ~ Phương thức**

Ví dụ 1 đối tượng hotel

```
var hotel = {
```

```
    name: 'Quay',  
    rooms: 40,  
    booked: 25,  
    gym: true,  
    roomTypes: ['twin', 'double', 'suite'],
```

```
    checkAvailability: function() {  
        return this.rooms - this.booked;  
    }
```

```
};
```

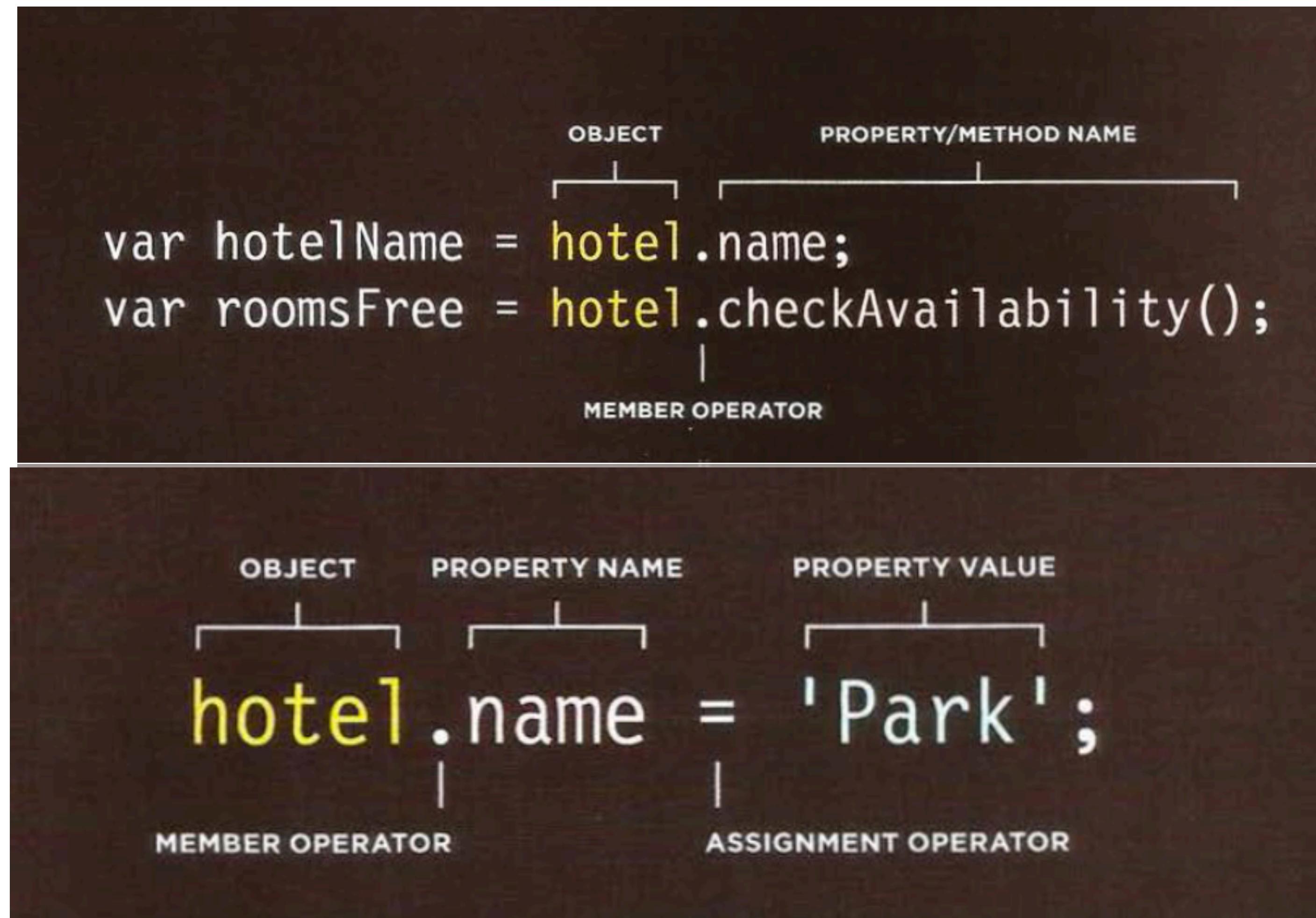


PROPERTIES
These are variables

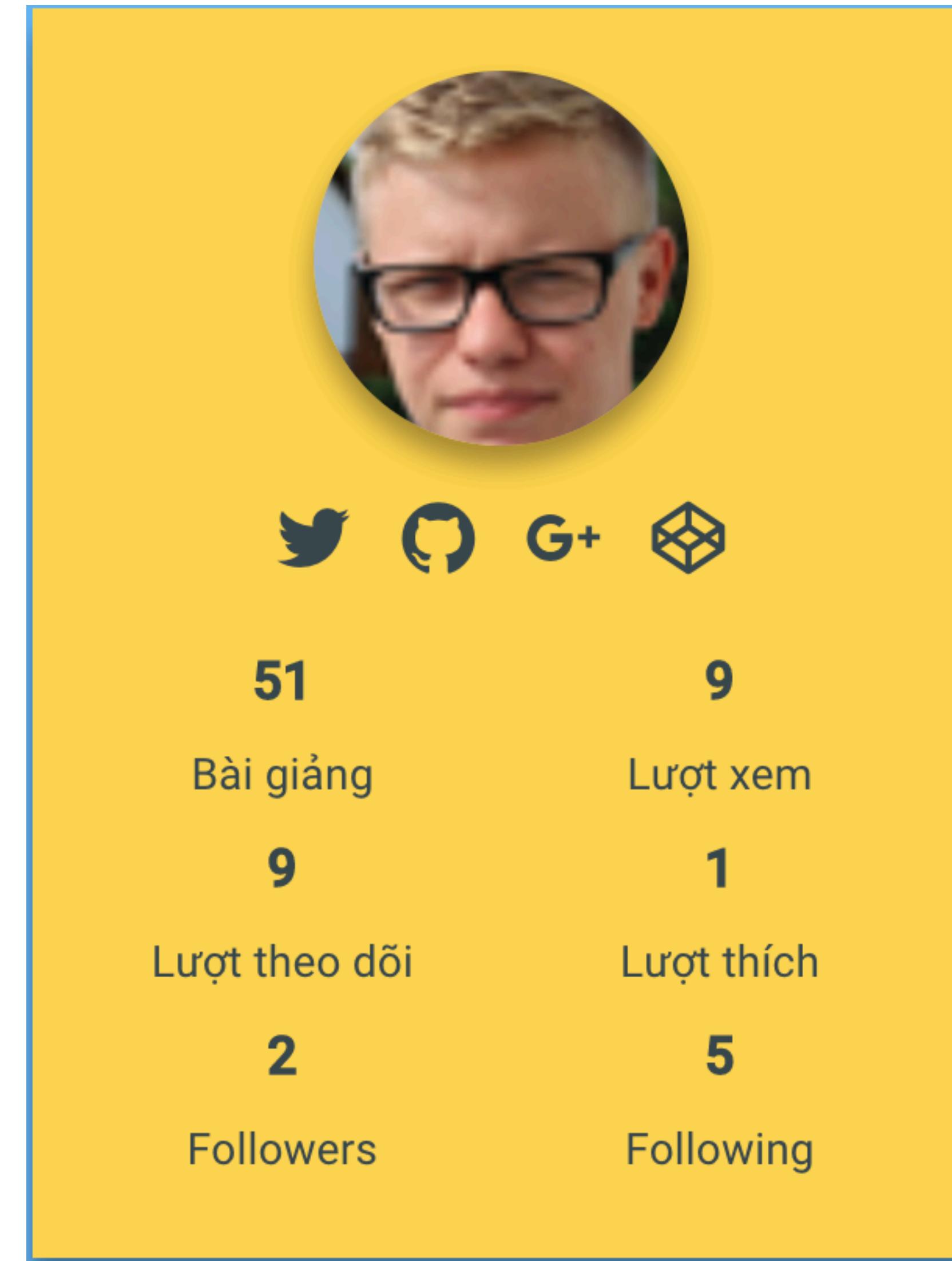
METHOD
This is a function

PROPERTIES:	KEY	VALUE
	name	string
	rooms	number
	booked	number
	gym	Boolean
	roomTypes	array
METHODS:	checkAvailability	function

Lấy & cập nhật thuộc tính



TỰ LÀM



JS OOP - SIMPLE OBJECT

YÊU CẦU DỰ ÁN

- * Xây dựng đối tượng để lưu trữ thông tin
- * Lấy dữ liệu từ mảng dữ liệu để đổ thông tin vào đối tượng vừa tổ chức
- * Viết sự kiện lấy dữ liệu
- * Các link social được nhập vào (sử dụng **setAttribute** để thiết lập cho **href, src**)

DỰ ÁN QUẢN LÝ NHÂN VIÊN

Quản lý nhân viên

Họ *

Vui lòng nhập họ

Tên*

Vui lòng nhập tên

Mã nhân viên*

Vui lòng nhập MSNV

04/01/2018 15

Chọn chức vụ*

Vui lòng chọn chức vụ

Thêm Nhân Viên

TÌM NHÂN VIÊN: **Tìm ngay**

Họ	Tên	MSNV	Ngày làm	Chức vụ	Lương	Thao tác
44	444	44	04/01/2018	Sếp	1050 \$	Sửa Xóa

JS OOP - SIMPLE OBJECT

YÊU CẦU DỰ ÁN

* Xây dựng hệ thống quản lý sinh viên với các yêu cầu sau:

- Thêm sinh viên mới
- Kiểm tra dữ liệu hợp lệ
- Hiển thị danh sách sinh viên đã thêm vào
- Tìm kiếm học sinh theo các tiêu chí
- Sửa thông tin sinh viên
- Xóa sinh viên

YÊU CẦU DỰ ÁN - NÂNG CAO

- *Phân trang cho table nếu lớn > 10*
- *Cho phép sắp xếp các column trong bảng dữ liệu*

KIẾN THỨC SAU DỰ ÁN

- * Kiểm tra dữ liệu nhập hợp lệ
- * Class - Prototype
- * Hàm khởi tạo
- * Thể hiện của đối tượng
- * Phân tích đối tượng trong dự án
- * Sơ đồ lớp
- * Cập nhật dữ liệu - giao diện
- * Hàm **indexOf** trong mảng
- * Hàm **splice** trong mảng
- * Các cách viết Event

KIỂM TRA HỢP LỆ

- * Kiểm tra bắt buộc nhập
- * Kiểm tra tất cả là số
- * Kiểm tra tất cả là kí tự
- * Kiểm tra email hợp lệ
- * Kiểm tra chiều dài chuỗi hợp lệ

Hàm kiểm tra trường dữ liệu **bắt buộc nhập**



```
function required()
{
    var empt = document.forms["form1"]["text1"].value;
    // hoặc lấy từ document.getElementById();
    if (empt == ""){
        alert("Please input a Value");
        return false;
    }else {
        alert('Code has accepted : you can try another');
        return true;
    }
}
```

56

Hàm kiểm tra trường dữ liệu **tất cả là kí tự**

```
function allLetter(inputtxt)
{
    var letters = /^[A-Za-z]+$/;
    if(inputtxt.value.match(letters))
    {
        alert('Your name have accepted : you can try another');
        return true;
    }
    else
    {
        alert('Please input alphabet characters only');
        return false;
    }
}
```

Hàm kiểm tra trường dữ liệu tất cả là số

68

```
function allNumeric (inputtxt)
{
    var numbers = /^[0-9]+$/;
    if(inputtxt.value.match(numbers))
    {
        alert('Your Registration number has accepted....');
        document.form1.text1.focus();
        return true;
    } else {
        alert('Please input numeric characters only');
        document.form1.text1.focus();
        return false;
    }
}
```

58

Hàm kiểm tra trường dữ liệu Email hợp lệ

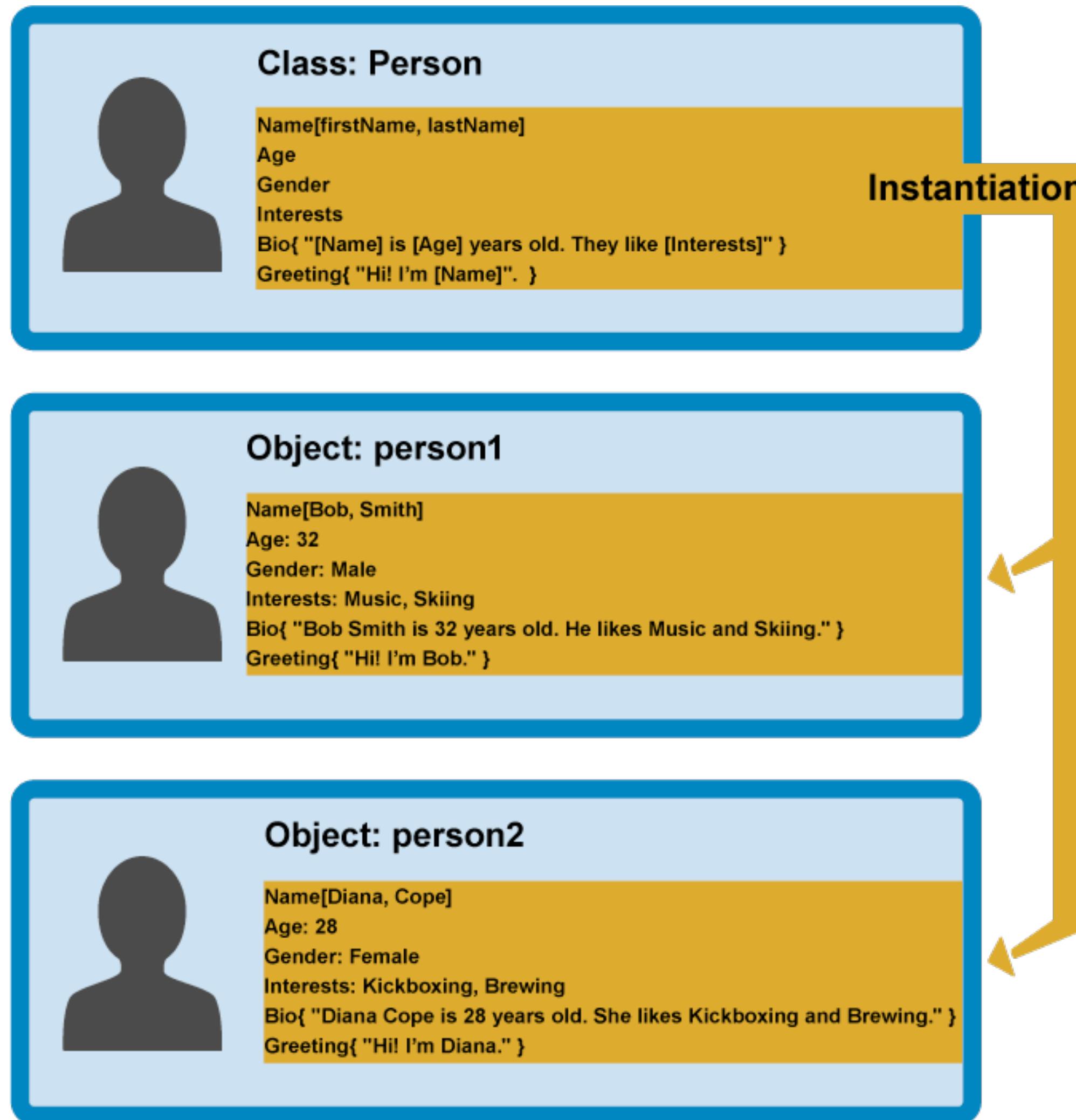
```
function checkEmail()
{
    var mailformat = /^[^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/;
    if(inputText.value.match(mailformat)){
        document.form1.text1.focus();
        return true;
    }else{
        alert("You have entered an invalid email address!");
        document.form1.text1.focus();return false;
    }
}
```

Hàm kiểm tra trường dữ liệu **chiều dài chuỗi** nhập vào



```
function stringlength(inputtxt, minlength, maxlength)
{
    var field = inputtxt.value;
    var mnlen = minlength;
    var mxlen = maxlength;
    if(field.length<mnlen || field.length> mxlen){
        alert("Please input the userid between " +mnlen+ " and " +mxlen+ " characters");
        return false;
    } else {
        alert('Your userid have accepted.');
        return true;
    }
}
```

JS OOP - CLASS - PROTOTYPE - TEMPLATE

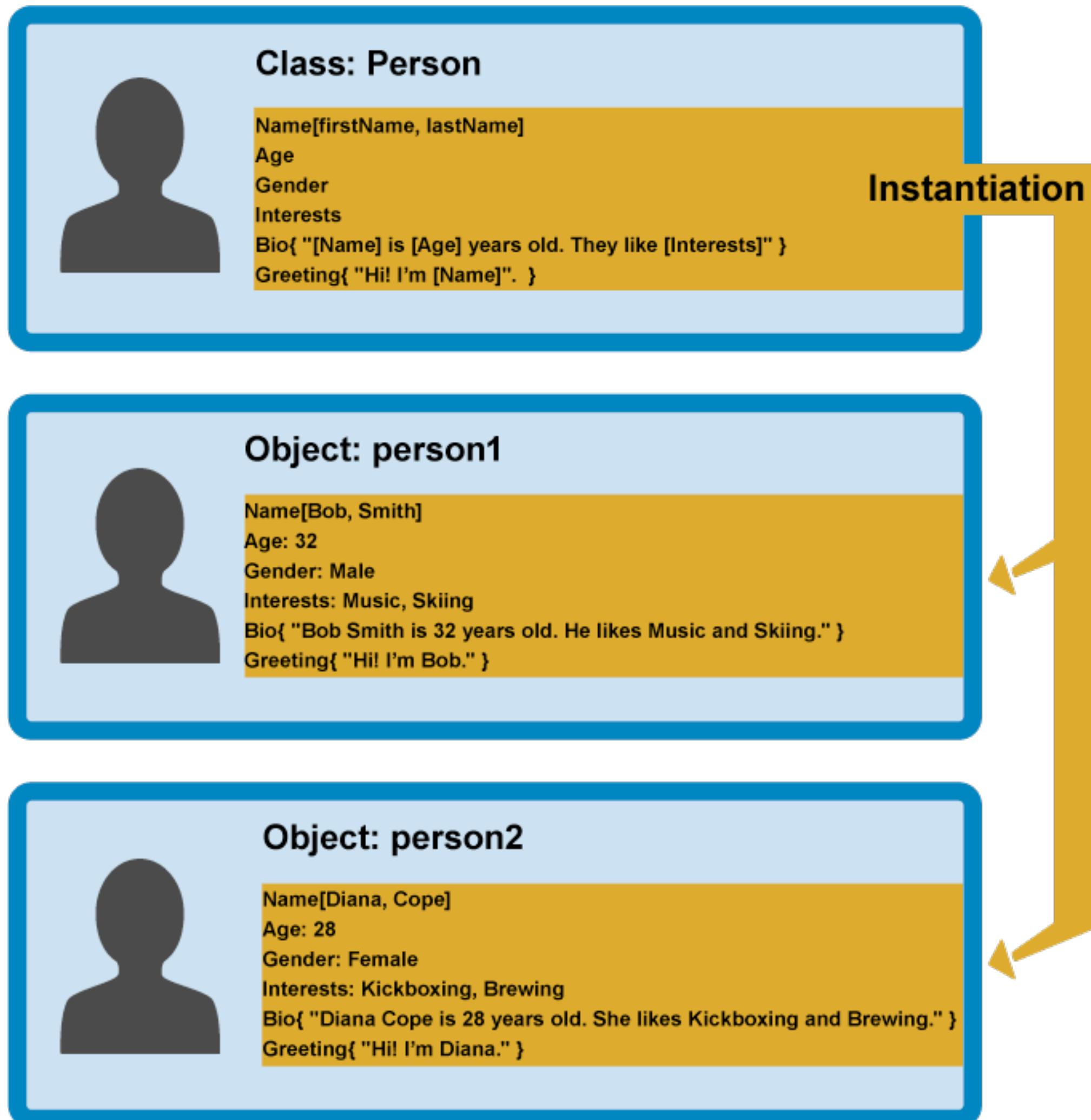


```
1 function Person(first, last, age, gender, interests) {  
2     this.name = {  
3         'first': first,  
4         'last' : last  
5     };  
6     this.age = age;  
7     this.gender = gender;  
8     this.interests = interests;  
9     this.bio = function() {  
10        alert(this.name.first + ' ' + this.name.last + ' is ' +  
11    );  
12    this.greeting = function() {  
13        alert('Hi! I\'m ' + this.name.first + '.');  
14    };  
15 }
```

- Tạo một template có thuộc tính và phương thức
- This: Từ khóa ám được sử dụng thay thế tên đối tượng để ám chỉ thuộc tính hoặc phương thức thuộc về đối tượng
- Object là nhóm các biến và các hàm để tạo ra một mô hình trong thực tế

61

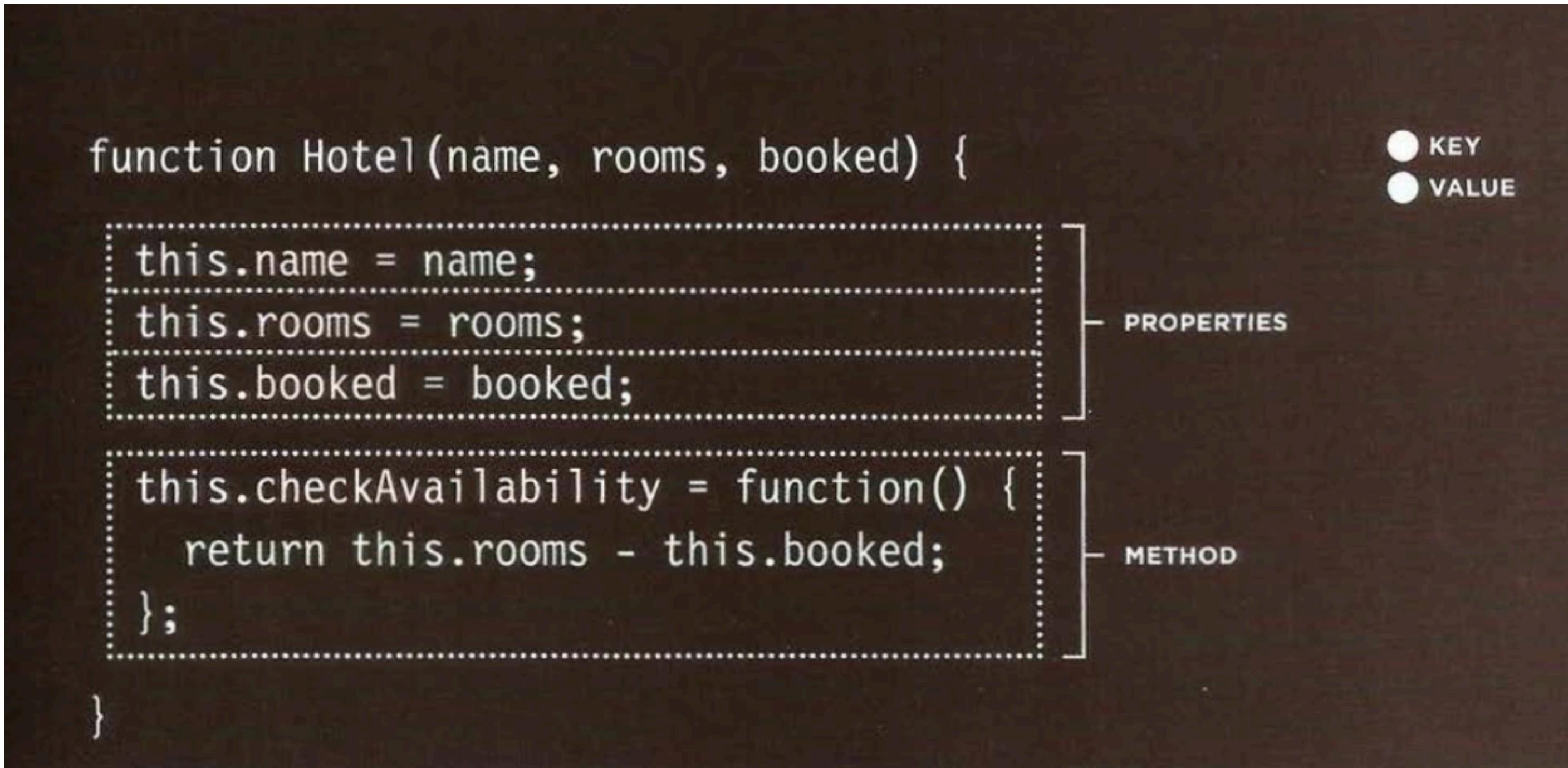
THỂ HIỆN (INSTANCE) CỦA LỚP ĐỐI TƯỢNG



```
1 function Person(first, last, age, gender, interests) {  
2     this.name = {  
3         'first': first,  
4         'last' : last  
5     };  
6     this.age = age;  
7     this.gender = gender;  
8     this.interests = interests;  
9     this.bio = function() {  
10        alert(this.name.first + ' ' + this.name.last + ' is ' +  
11    );  
12    this.greeting = function() {  
13        alert('Hi! I\'m ' + this.name.first + '.');  
14    };  
15 }
```

```
var person1 = new Person('Bob', 'Smith', 32, 'male',  
['music', 'skiing']);
```

THỂ HIỆN (INSTANCE) CỦA LỚP ĐỐI TƯỢNG



- Thể hiện của một lớp đối tượng : giá trị thuộc tính khác nhau
- Từ khóa **new** để tạo ra một thể hiện của lớp đối tượng
- Hàm sau từ khóa **new** gọi là *hàm khởi tạo*

PROTOTYPE - BỔ SUNG THUỘC TÍNH & PHƯƠNG THỨC

```
function Person(first, last, age, eyecolor) {  
    this.firstName = first;  
    this.lastName = last;  
    this.age = age;  
    this.eyeColor = eyecolor;  
}
```

```
Person.prototype.nationality = "English";
```

```
function Person(first, last, age, eyecolor) {  
    this.firstName = first;  
    this.lastName = last;  
    this.age = age;  
    this.eyeColor = eyecolor;  
}
```

```
Person.prototype.name = function() {  
    return this.firstName + " " + this.lastName;  
};
```

PHÂN TÍCH DỰ ÁN - SƠ ĐỒ LỚP

Phân tích hướng đối tượng

The screenshot shows a web-based application interface for managing employees. At the top, there is a modal window titled "Quản lý nhân viên" (Employee Management) containing fields for "Họ*" (Last Name), "Tên*" (First Name), "Mã nhân viên*" (Employee ID), and "Chọn chức vụ*" (Select Position). Below the modal is a search bar with a dropdown menu and a text input field. At the bottom, there is a table with columns: Họ (Last Name), Tên (First Name), MSNV (Employee ID), Ngày làm (Date of Birth), Chức vụ (Position), Lương (Salary), and Thao tác (Actions). One row in the table is highlighted.

Nhân Viên

maNV
ho
ten
email
ngaySinh
chucVu
mangThongTin

tinhLuong();

Danh Sách Nhân Viên

mangNV

xuatLuong();
themNhanVien();
xoaNhanVien(ma);
suaNhanVien(ma);
timNVTheoTen(ten);
timNVTheoMa(ma);

Các bước thực hiện - Thêm dữ liệu

★Bước 0: Tao cac lop du lieu

★Bước 1: Thêm dữ liệu vào Danh sách

★Bước 1.1: Kiem tra du lieu hop le

★Bước 1.2: Viet them cac phuong thuc xu ly cho cac lop doi tuong

★Bước 1.3: Xu ly su kien click them

★Bước 2: Hiển thị danh sách vào table

- ✓ Tạo từng dòng table dựa vào số lượng thuộc tính (Nếu hiển thị tất cả thuộc tính, nếu hiển thị ít hơn thì bỏ qua thuộc tính đó khi chạy lặp)
- ✓ Lặp trên danh sách và tạo ra các dòng
- ✓ Bổ sung động cột STT
- ✓ Bổ sung động 3 nút : Xóa, Sửa, Cap Nhat
- ✓ Tạo ID cho 3 nút gắn với ID của MÃNV để xử lý cho 3 nút tương ứng với Nhân viên
- ✓ Tạo hàm xử lý động cho 3 nút

Các bước thực hiện - Delete

★ Viết hàm Delete

- ✓ Lấy ID của nút
- ✓ Dùng hàm split để tách chuỗi → lấy ID của Mã NV
- ✓ Gọi hàm Xóa nhân viên theo mã

Các bước thực hiện - nút Edit - cập nhật

- ✓ Lấy ID của nút
- ✓ Dùng hàm split để tách chuỗi —> lấy ID của Mã NV
- ✓ Gọi hàm tìm nhân viên —> lấy nhân viên
- ✓ Lấy row hiện hành chưa nút click, sử dụng : **parentNode** và **rowIndex**
(element.parentNode.parentNode.rowIndex)
- ✓ Tạo các textbox trong các cell của table tại row hiện hành & Gán dữ liệu cho Textbox —>
Loop các thuộc tính của NVvien -> truy xuất đến cell —> tạo textbox & gán dữ liệu
- ✓ Chuyển sang nút Cập nhật (gọi hiện nút cập nhật - ban đầu đã có)
- ✓ Sự kiện nút cập nhật: Lấy giá trị các textbox —> cập nhật vào danh sách
- ✓ Xóa table và tạo lại các row theo danh sách Hoặc lặp lại và **gánInnerHTML cho cell bằng giá trị mới nhập vào để tăng performance**

Các bước thực hiện

- ★Nâng cao: Xay dung toan bo giao dien PRO hon
- ★Viết hàm Phân trang nếu lớn 10 item, viết hàm sort table
- ★Luôn luôn nhớ rằng, mọi thứ đều xuất phát từ dữ liệu, table hay control giao diện chỉ để hiển thị mà thôi —> mọi thao tác đều xử lý ở DỮ LIỆU —> cập nhật lên giao diện —> Đảm bảo dữ liệu luôn toàn vẹn từ nhiều nơi

Mảng trong JS

```
var myArray = ["one", "two", 3, 4, "five", "six", 7, 8, 9];
console.log(myArray.sort());
console.log(myArray.reverse());
myArray.push("ten");
console.log(myArray);
myArray.pop();
console.log(myArray);
var a = myArray.shift();
console.log(a);
console.log(myArray);
myArray.unshift("ten");
console.log(myArray);
myArray.splice(1, 2);
console.log(myArray);
myArray.splice(3, 0, "added 1", "added 2");
console.log(myArray);
```

Mảng trong JS

```
var cars = new Array();  
var cars = new Array('Honda', 'Toyota', 'BMW');  
cars[3] = "Dodge";
```

```
var people = [];  
var people = ['Nam', 'Lien', 'Nguyen'];
```

```
document.write(cars[0]);  
document.write(cars.length);  
cars.indexOf("Toyoto");
```

```
var myArray = ["Song", 4, 5, true, false, "course"];  
console.log (myArray);  
myArray.sort (); // số trước, text alphabe,...
```

CHO DỰ ÁN QUẢN LÝ KHÓA HỌC

YÊU CẦU DỰ ÁN

- * Xây dựng hệ thống quản lý các khóa học online dữ liệu từ Server với các yêu cầu sau đây:

Phân hệ Admin

- * Hệ thống cho phép Admin Quản trị các khóa học (Thêm, Xóa, Sửa)
- * Hệ thống cho phép Admin quản lý danh sách học viên đăng ký
- * Dashboard có sử dụng các charts các số liệu về số khóa học, số học viên

Phân hệ Người dùng

- * Hệ thống trang chủ giới thiệu về trung tâm, khóa học
- * Trang danh sách các khóa học
- * Trang chi tiết khóa học
- * Trang đăng ký, đăng nhập thành viên
- * Cho phép người dùng sửa thông tin đăng ký

ĐỌC THÊM VÀ LÀM THÊM

Bài tập 1:

Cho 2 mảng:

$$a = [3, -6, 8, -9, -4, 5, 12]$$

$$b = [4, 6, 12, 14, 8]$$

- 1) Tính tổng các số trong mảng a và mảng b
- 2) Tìm phần tử âm lớn nhất trong mảng a. Xuất giá trị và chỉ số tại vị trí đó
- 3) Tính tổng các số lẻ trong mảng a và b
- 4) Tìm số lớn nhất trong mảng a và b và xuất chỉ số tại đó.

Cho 5 sinh viên có điểm số và tên theo thứ tự (Tên, Toán, Lý, Hóa) như sau:

```
sv1 = ["song", 9, 9, 9];
```

```
sv2 = ["le", 3, 5, 7];
```

```
sv3 = ["quang", 7, 8, 6];
```

```
sv4 = ["Luc", 7, 6, 8];
```

```
sv5 = ["Tưởng", 3, 2, 5];
```

```
sv = [];
```

```
sv.push(sv1); // -> mảng sinh viên.
```

```
dbt = [];
```

```
// duyệt trên mảng sinh viên
```

```
svTemp = sv[i][j];
```

```
// duyệt mảng điểm và tên : 1 -> svTemp.length
```

Thực hiện và xuất ra

1) Điểm trung bình của từng sinh viên và tên của họ. (cho rằng Toán, lý, hóa cùng hệ số)

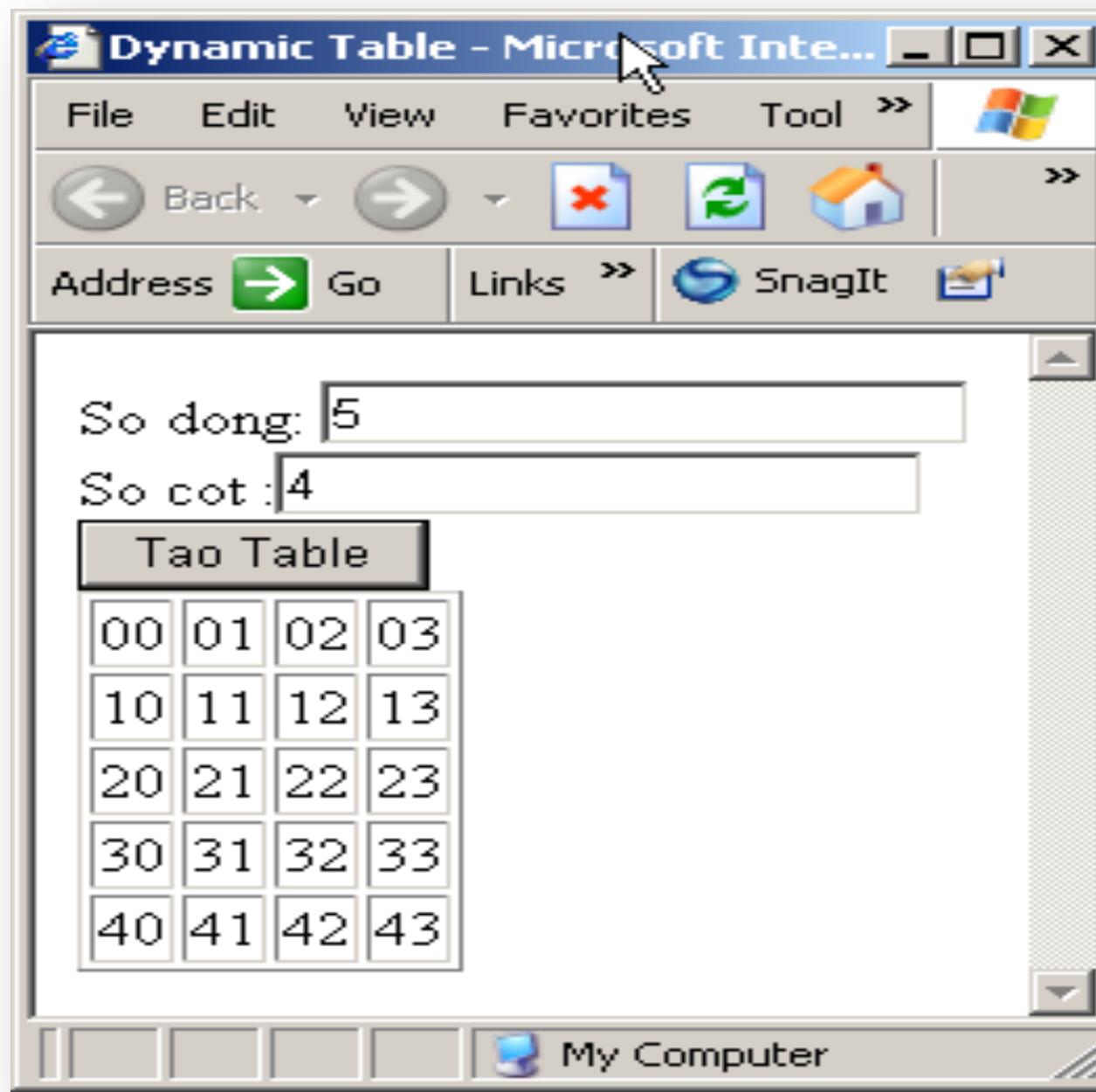
2) Tìm SV có ĐTB cao nhất.

3) Xếp loại cho SV biết (≥ 9 : Xuất Sắc, $<9 \&& \geq 8$ -> Giỏi, $<8 \&& \geq 7$, Khá, $<7 \&& \geq 6$

\rightarrow TBKha , $<6 \&& \geq 5$: TB, còn lai Yếu)

Ví dụ tạo table động

- Viết trang web cho phép tạo table có số dòng, số cột do người dùng nhập vào.

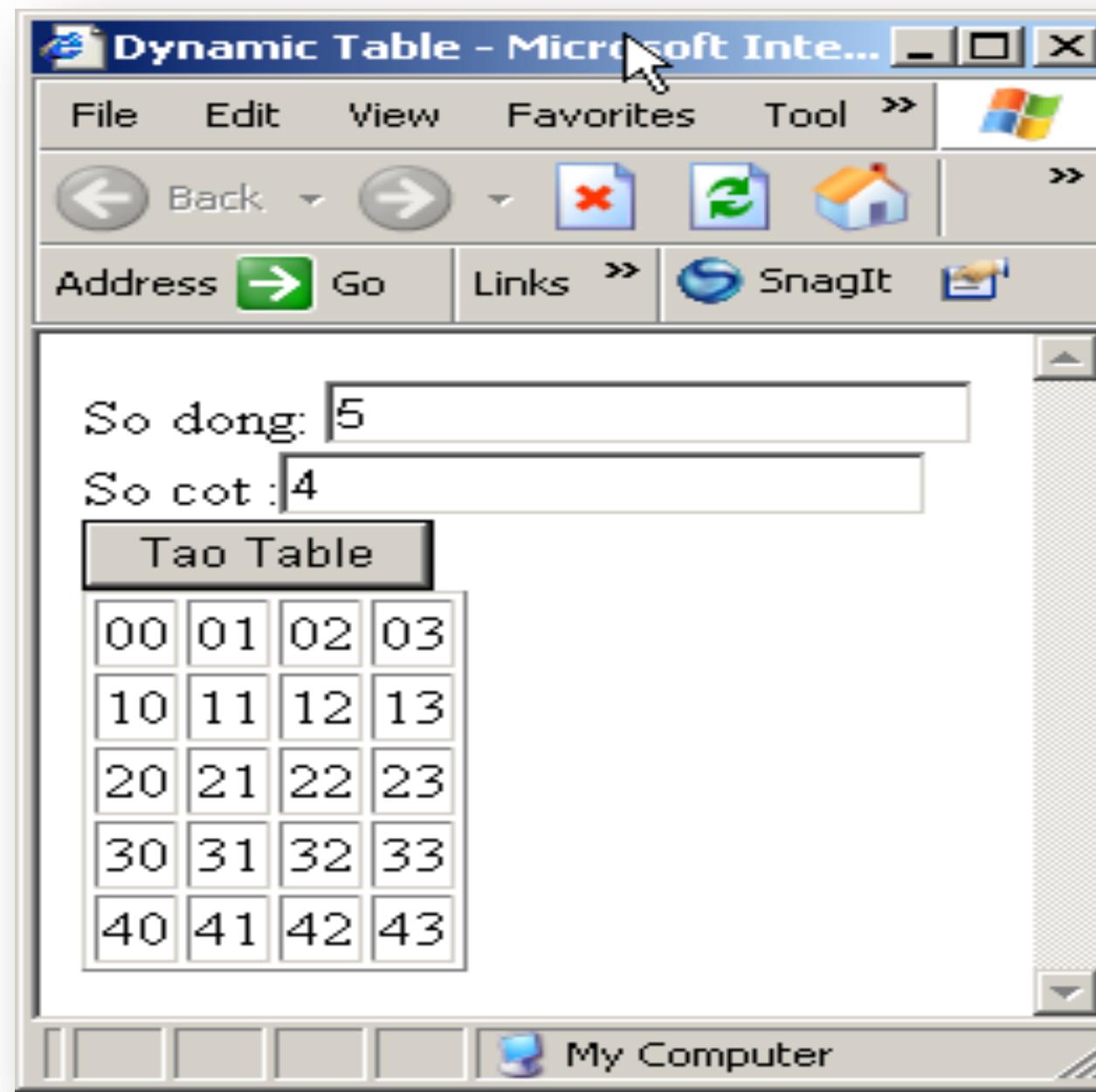


- + `Document.createElement(...)`: Tạo một đối tượng thẻ DOM HTML
- + `Object.appendChild(...)`: Thêm một đối tượng thẻ DOM HTML như là nút con.
- + Đối tượng Number trong Javascript
 - `parseInt(...)`
 - `parseFloat(...)`



Tạo dropdown động bằng js

- Viết trang web cho phép tạo table có số dòng, số cột do người dùng nhập vào.



- + `Document.createElement(...)`: Tạo một đối tượng thẻ DOM HTML
- + `Object.appendChild(...)`: Thêm một đối tượng thẻ DOM HTML như là nút con.
- + Đổi tượng Number trong Javascript
 - `parseInt(...)`
 - `parseFloat(...)`



Đối tượng Date trong Javascript



// Đối tượng thời gian hiện tại

```
var d = new Date();
```

```
d.getDate();
```

// getDate() lấy ngày (1 - 31)

```
d.getDay();
```

// getDay() lấy ngày trong tuần (0-6)

```
d.getFullYear();
```

// getFullYear() lấy năm đầy đủ (YYYY)

```
d.getYear();
```

// getYear() lấy 2 số cuối của năm (YY)

```
d.getHours();
```

// getHours() lấy số giờ (0 - 23)

```
d.getMilliseconds();
```

// getMiliSeconds() lấy số mili giây (0 - 999)

```
d.getMinutes();
```

// getMinutes() lấy số phút (0 - 59)

```
d.getMonth();
```

// getMonth() lấy tháng (0 - 11)

```
d.getSeconds();
```

// getSeconds() lấy số giây (0 - 59)

```
d.getTime();
```

// getTime() thời gian đã được chuyển đổi sang dạng mili giây



Các phương thức xử lý chuỗi

1. Tìm kiếm chuỗi con

Có ba phương thức thường dùng để tìm kiếm chuỗi con trong JavaScript như sau:

`indexOf()`

`lastIndexOf()`

`search()`

Để tìm kiếm chuỗi con, chúng ta sử dụng phương thức **`String.indexOf(str)`**, trong đó **`str`** là chuỗi con và **`String`** là chuỗi cha.

Phương thức này sẽ trả về kết quả là **vị trí xuất hiện đầu tiên của chuỗi (bắt đầu là vị trí 0)**, nếu **không** tìm thấy chuỗi con thì nó sẽ trả về **-1**.



Các phương thức xử lý chuỗi

Phương thức `lastIndexOf()`

Trường hợp nếu chuỗi con xuất hiện nhiều lần trong chuỗi cha thì kết quả cũng trả về vị trí xuất hiện của chuỗi con đầu tiên.

Do đó, để lấy vị trí của chuỗi con cuối cùng trong chuỗi cha thì chúng ta phải sử dụng phương thức `String.lastIndexOf(str)`.

Phương thức này sẽ trả về vị trí xuất hiện của chuỗi con cuối cùng và trả về -1 nếu không tìm thấy.



Các phương thức xử lý chuỗi



Phương thức search()

Ngoài hai phương thức trên, bạn có thể sử dụng phương thức **string.search(str)** để tìm kiếm, tác dụng của nó cũng giống như phương thức **string.indexOf(str)**.

2. Cắt chuỗi con

Nếu bạn muốn cắt một chuỗi con từ chuỗi cha thì có thể sử dụng ba phương thức sau:

slice(start, end)

substring(start, end)

substr(start, length)

Ghi chú: Tất cả các vị trí của chuỗi đều bắt đầu từ 0, vì vậy khi tính toán vị trí phải thật cẩn thận.



Các phương thức xử lý chuỗi



Phương thức slice()

Phương thức **slice()** có *hai tham số* truyền vào:

start: vị trí bắt đầu

end: vị trí kết thúc

Nếu tham số truyền vào là số âm thì nó sẽ tính ngược lại, nghĩa là nó sẽ đếm từ cuối lên.

Nếu bạn chỉ truyền một tham số đầu tiên thì nó sẽ tự hiểu vị trí end là vị trí cuối cùng.



Các phương thức xử lý chuỗi



Phương thức substring()

Phương thức **substring()** có cách sử dụng giống với phương thức **slice()**, tuy nhiên tham số truyền vào phương thức **substring()** phải luôn luôn **lớn hơn 0**.

Phương thức substr()

Phương thức **substr()** có hai tham số là **start** và **length**, trong đó **start** là vị trí bắt đầu và **length** là số ký tự muốn lấy bắt đầu từ vị trí **start**. Nếu bạn truyền tham số **start** là số âm thì nó sẽ tính từ **cuối trở lên**, còn tham số **length** phải luôn luôn là **số dương**.



Chuỗi, hộp thoại trong js

https://www.youtube.com/watch?v=QXxvs_INHv8



DOM + thay đổi CSS

https://www.youtube.com/watch?v=ez_zC0dUP9I



DOM Validation

Bài 52 + 53 + 54 trong khóa học



BÀI TẬP JS TỔNG HỢP



CHO BÀI TẬP ĐĂNG KÍ + ĐĂNG NHẬP -> VALIDATION + DOM

55 -> 58



BÀI TẬP JS TỔNG HỢP (máy tính cầm tay)



CHO BÀI TẬP ĐĂNG KÍ + ĐĂNG NHẬP -> VALIDATION + DOM

55 -> 58



Dom CSS - animation , hide, show, event



<http://nhatdesign.com/hoc-javascript/dom-css-trong-javascript.html>

http://www.w3schools.com/js/js_htmldom_css.asp

http://www.w3schools.com/js/js_htmldom_animate.asp

Thay đổi source , cái đèn : http://www.w3schools.com/js/js_intro.asp



Kiểm tra Form hợp lệ

Học lập trình trực tuyến tại myclass.vn

