Web Application Design

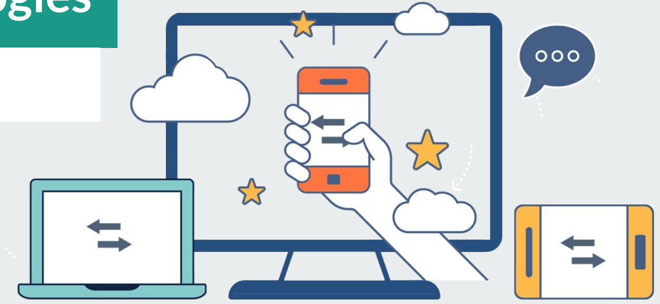
-Lecture 2-

Chapter 1 – **HTML, CSS, Javascript web technologies**

Part III: JavaScript language - Overview

Adil **CHEKATI**, PhD

adil.chekati@univ-constantine2.dz



Prerequisites

- ❏ Basic Understanding of Web Technologies.
- ❏ Programming Fundamentals

JavaScript language - Overview

Part III

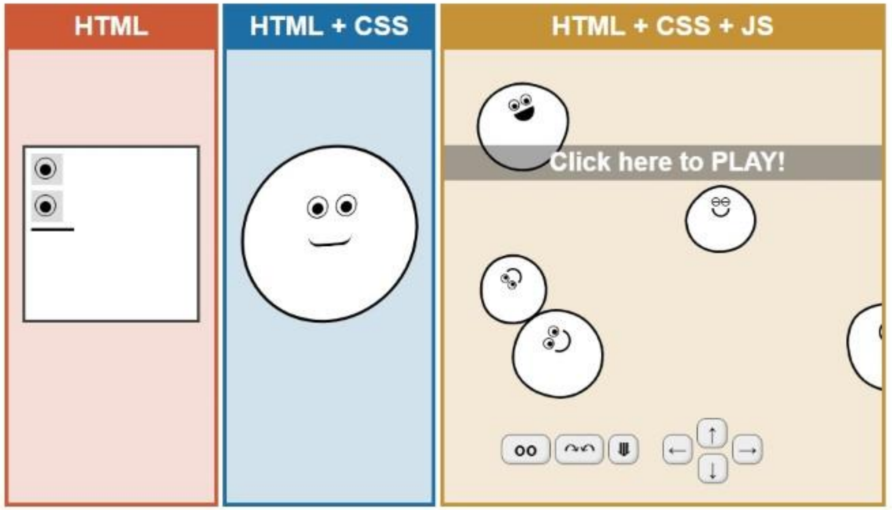
Objectives

- ➔ Comprehend the fundamental role and functionalities of JavaScript as a programming language for web applications.
- ➔ Utilize JavaScript for basic scripting tasks and interactivity within a web page.

1. JavaScript basics

A comprehensive tutorial on web technologies (HTML, CSS and JavaScript) can be found at: **https://developer.mozilla.org/fr/docs/Web**.

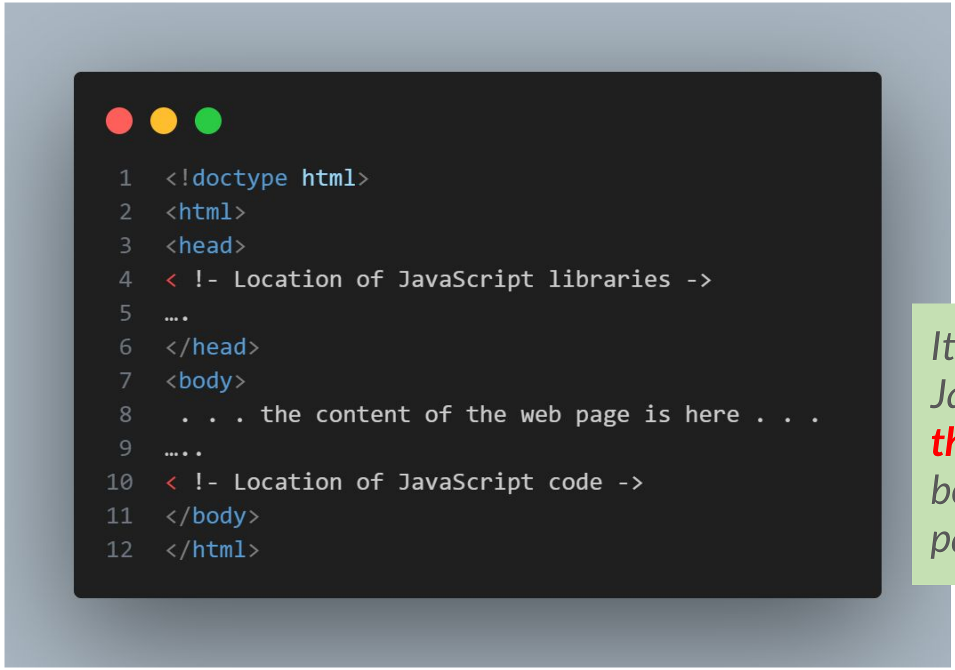
To see all the details of the JavaScript language with examples and detailed descriptions of all concepts.



(Please refer to this link for details omitted from this course).

1.1 Location of JavaScript code

In principle, **anywhere** in an HTML document, but there are some well-known patterns:



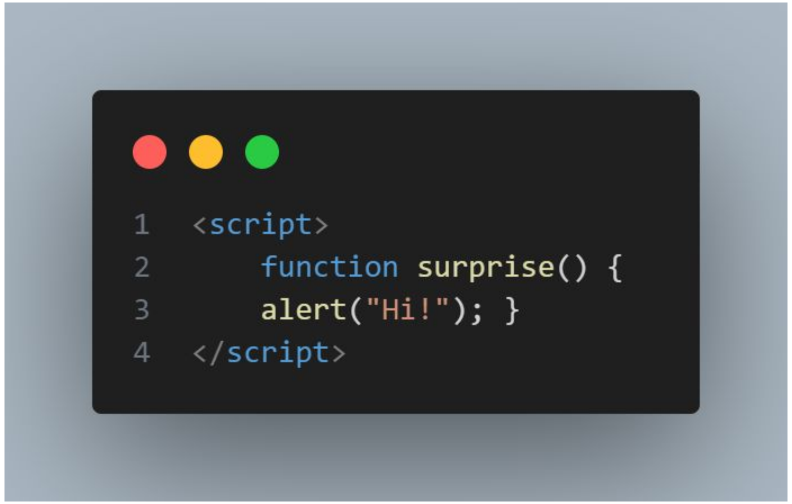
```
<!doctype html>
<html>
<head>
< !- Location of JavaScript libraries ->
….
</head>
<body>
 . . . the content of the web page is here . . .
…..
< !- Location of JavaScript code ->
</body>
</html>
```

*It's advisable to place the JavaScript code just **before the end** tag of the document body (</body>) to speed up page display.*

1.1 Location of JavaScript code

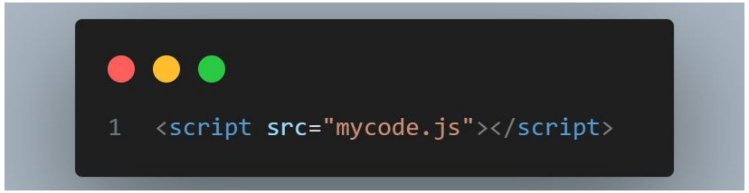
JavaScript code can either be located :

In the HTML file of the web page



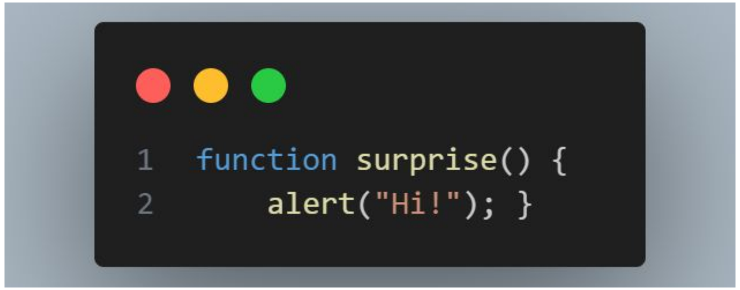
```
<script>
    function surprise() {
    alert("Hi!"); }
</script>
```

In an external file



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

In the file mycode.js:

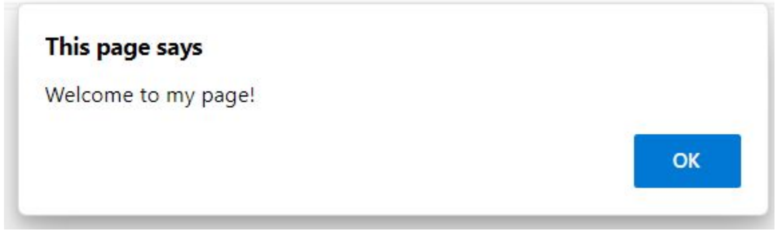


```
function surprise() {
     alert("Hi!"); }
```

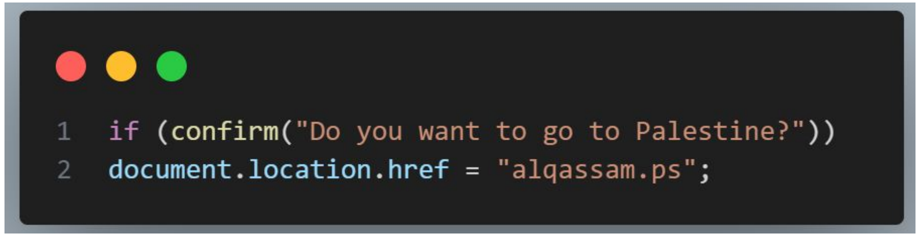
1.2 JavaScript dialogs

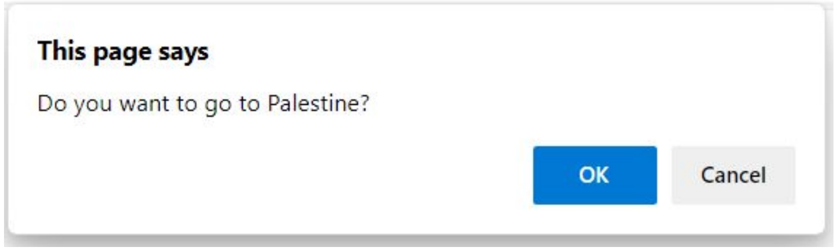
3 types :

alert() :
to display a message .
Ex : `alert("Welcome to my page!")`
Display :


This page says
Welcome to my page!

confirm():
to display a message with a decision.
Ex :

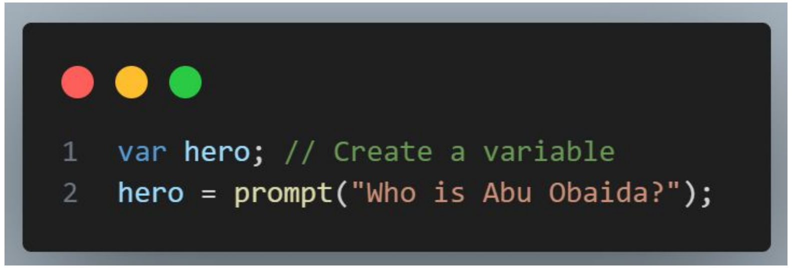

```
if (confirm("Do you want to go to Palestine?"))
document.location.href = "alqassam.ps";
```


This page says
Do you want to go to Palestine?

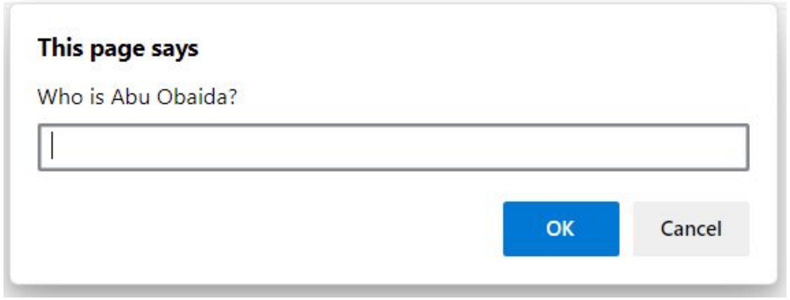
1.2 JavaScript dialogs

3 types :

prompt():
to display a message and receive data from the user
Ex:



```
var hero; // Create a variable
hero = prompt("Who is Abu Obaida?");
```

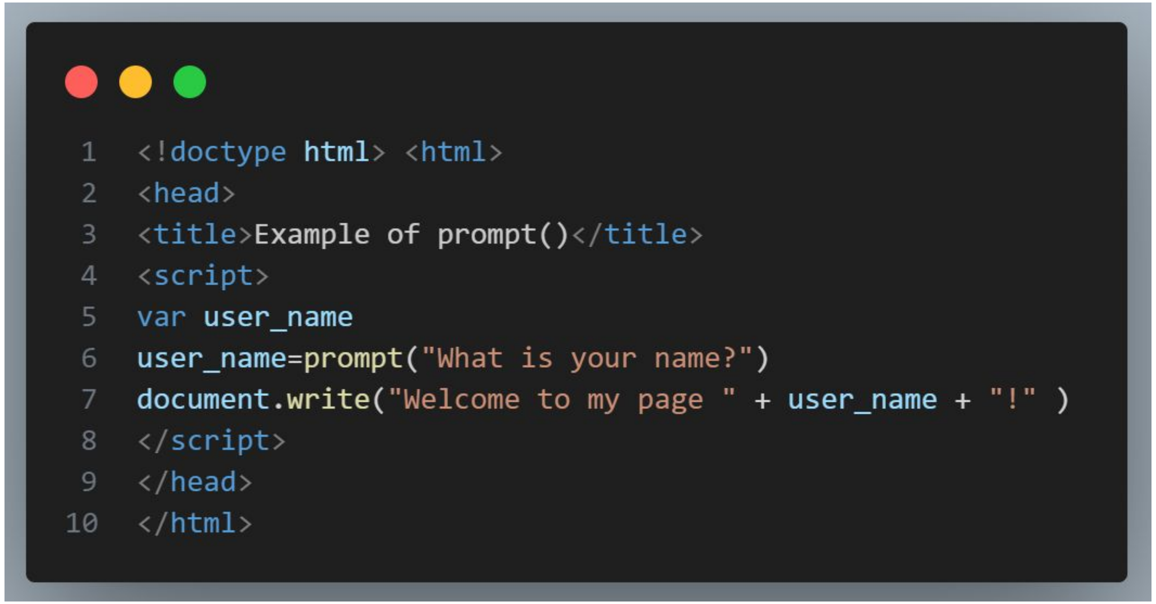


This page says

Who is Abu Obaida?

1.2 JavaScript dialogs

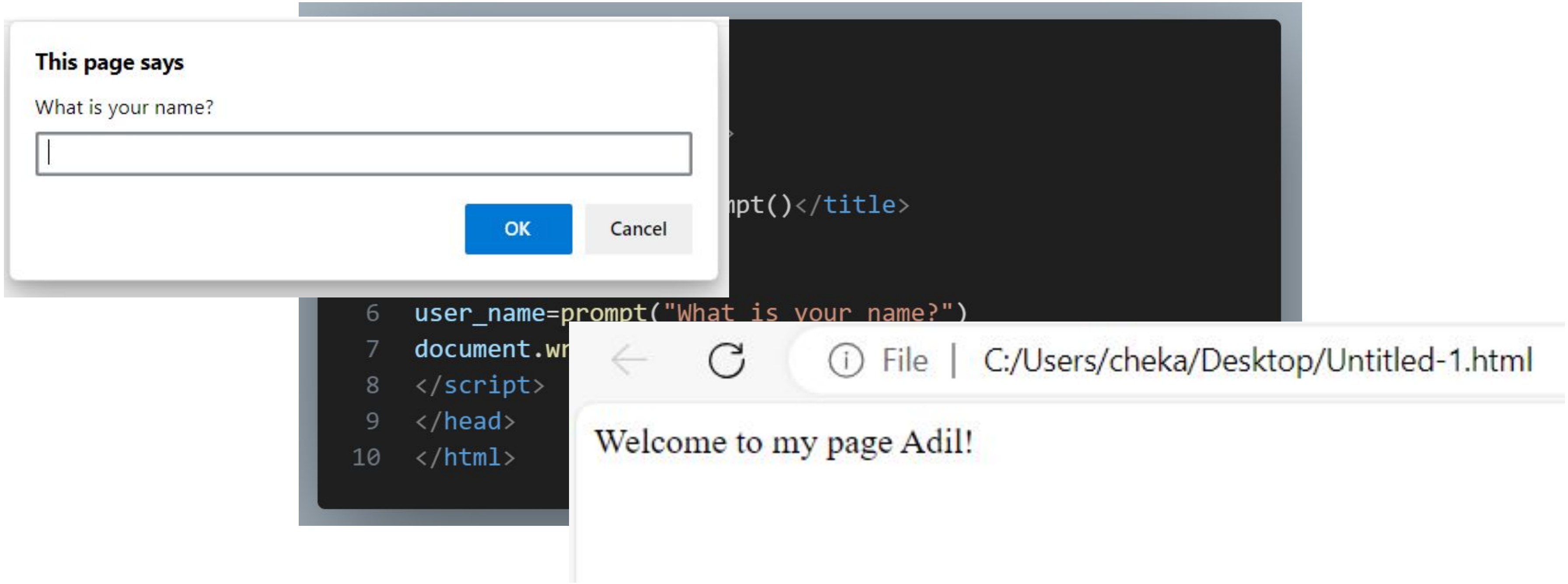
Example:



```
<!doctype html> <html>
<head>
<title>Example of prompt()</title>
<script>
var user_name
user_name=prompt("What is your name?")
document.write("Welcome to my page " + user_name + "!" )
</script>
</head>
</html>
```

1.2 JavaScript dialogs

JavaScript code can either be located :



This page says

What is your name?

OK Cancel

```
npt()</title>
user_name=prompt("What is your name?")
document.wr
</script>
</head>
</html>
```

File | C:/Users/cheka/Desktop/Untitled-1.html

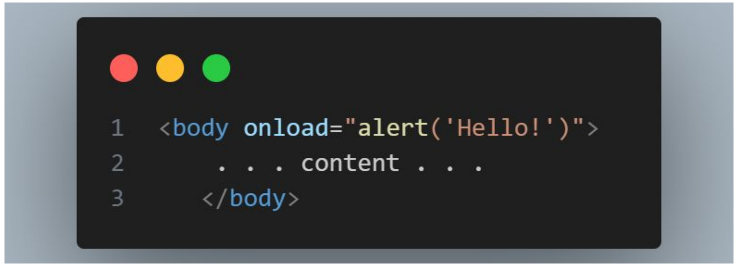
Welcome to my page Adil!

1.3 Events in JavaScript

An event indicates the occurrence of something important that needs to be dealt with.
For example: clicking on something, moving the mouse, pressing a key on the keyboard or a button...

Code can be scheduled to run when an event occurs, This is called **event processing.**
Examples of events: onclick, onmouseover, onmouseout, onmouseup, onmousedown, keypressed etc.

Example: the web page loading event: ***onload***



```
<body onload="alert('Hello!')">
    . . . content . . .
  </body>
```

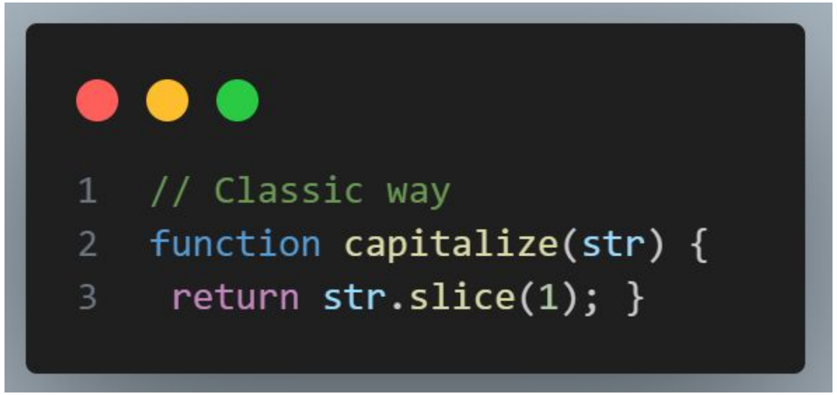
1.4 Functions in JavaScript

As in other programming languages, a **function** is a set of instructions executed when the function is called. It may or may not have parameters, return a result, make recursive calls to a function and so on.

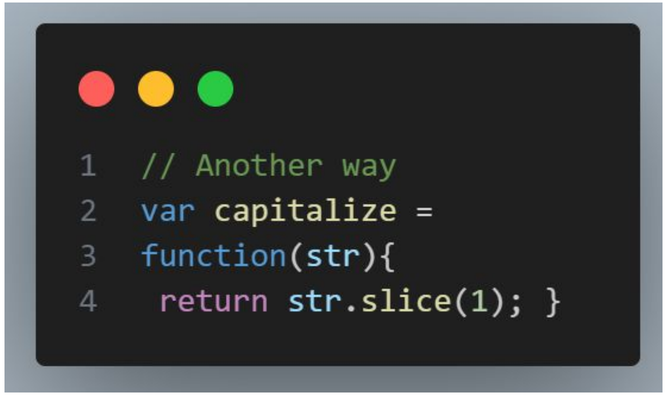
➔ In JavaScript, functions play a very important role. Almost all JavaScript code is written in the form of functions.
➔ A function that returns a result does so through the `return` keyword.
➔ The `return` keyword stops the execution of the function.
➔ A function can have another function as a parameter. known as a callBack function.

1.4 Functions in JavaScript

➔ A function can be declared in 2 ways in JS according to the following examples:



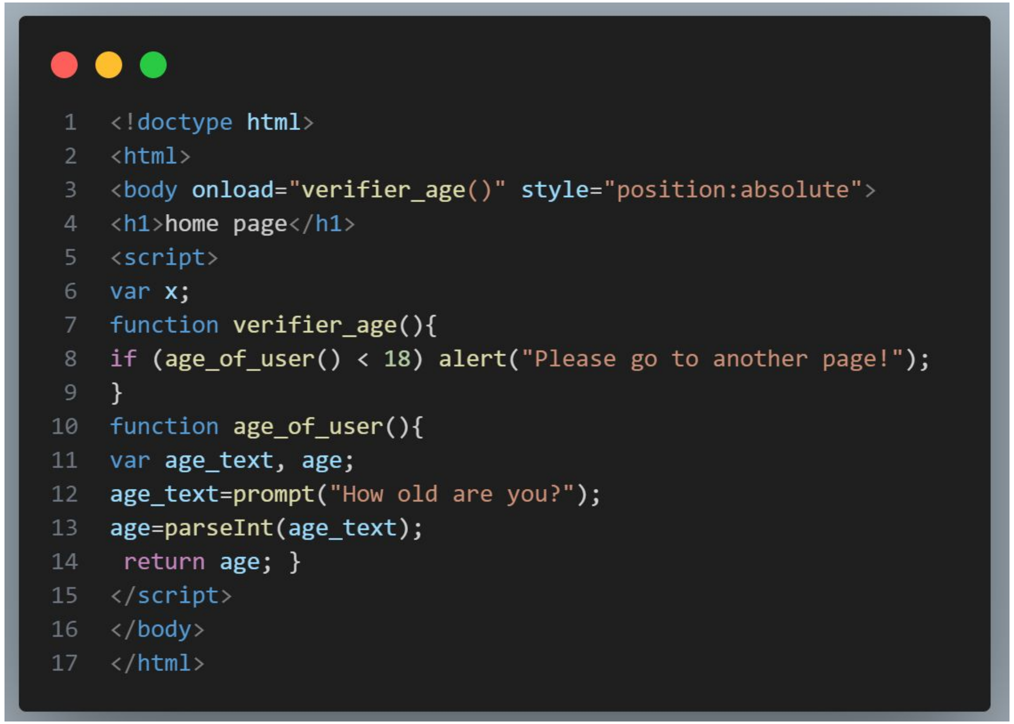
```
// Classic way
function capitalize(str) {
 return str.slice(1); }
```



```
// Another way
var capitalize =
function(str){
 return str.slice(1); }
```

1.4 Functions in JavaScript

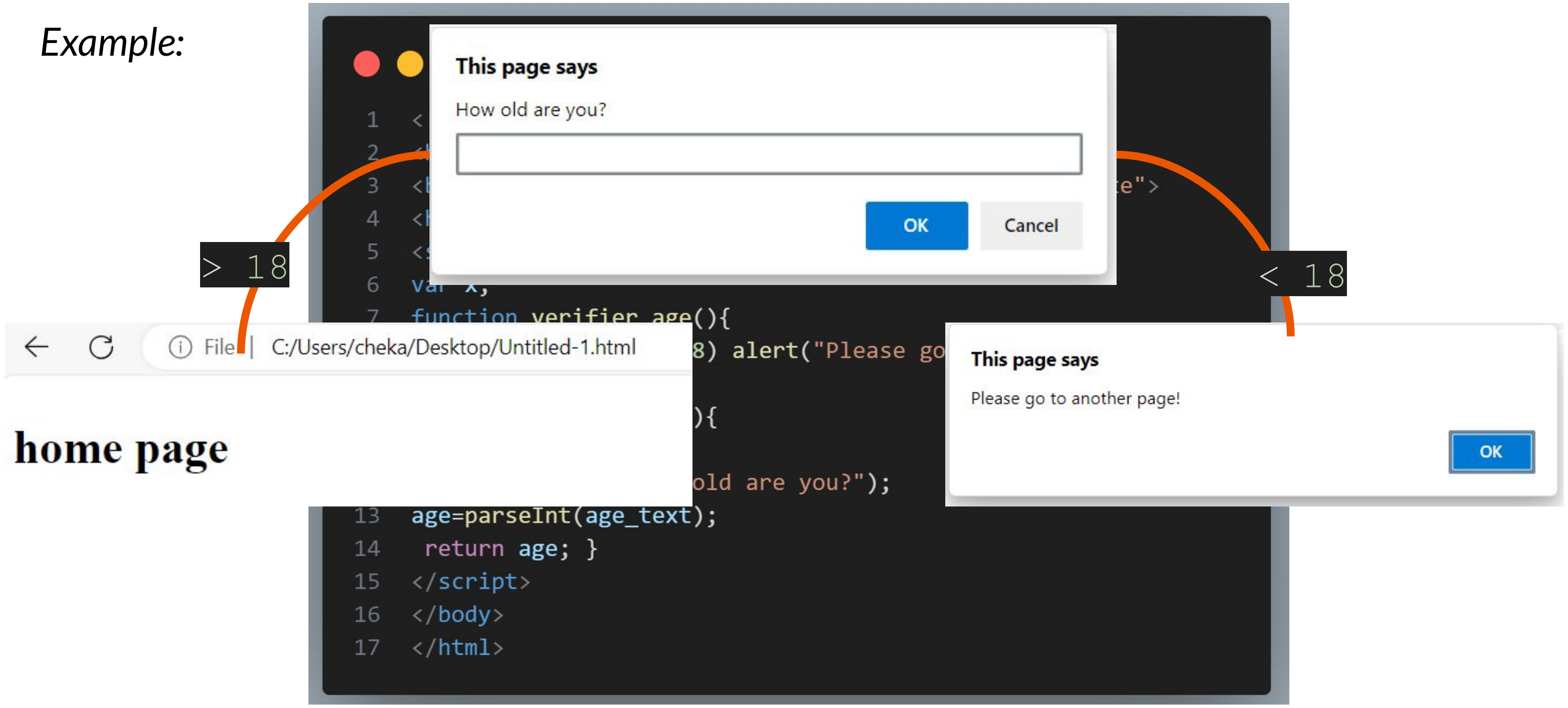
Example:



```
<!doctype html>
<html>
<body onload="verifier_age()" style="position:absolute">
<h1>home page</h1>
<script>
var x;
function verifier_age(){
if (age_of_user() < 18) alert("Please go to another page!");
}
function age_of_user(){
var age_text, age;
age_text=prompt("How old are you?");
age=parseInt(age_text);
 return age; }
</script>
</body>
</html>
```

1.4 Functions in JavaScript

Example:



```
<
<!
<b                                              e">
<h
<s
var x;
function verifier_age(){
                          8) alert("Please go
                          ){
                          old are you?");
age=parseInt(age_text);
 return age; }
</script>
</body>
</html>
```

This page says

How old are you?

This page says

Please go to another page!

File | C:/Users/cheka/Desktop/Untitled-1.html

home page

1.5 Local or global variables

Variables do not have to be declared (using the var keyword) in JavaScript.
But it is advisable to declare them in their functions to avoid certain errors.

➔ Variables declared inside a function are called local, and can only be used can only be used within that function
➔ Variables declared in the main program are called global and can be used both used both inside and outside functions.
➔ If two variables (one global and one local) share the same name, priority is given to the local variable within the function. for the local variable within its function.

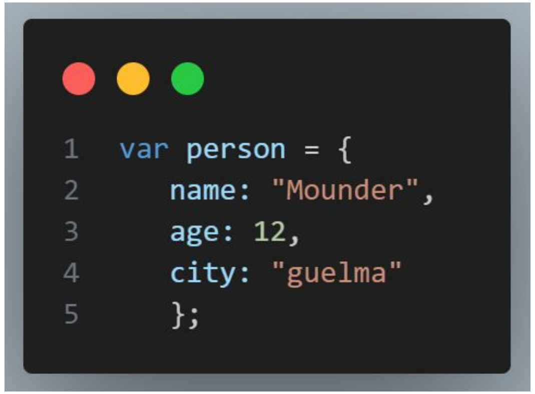
1.6 Objects in JavaScript

In JavaScript, everything is an object, including basic types and functions.
An object is defined as a set of pairs (`attribute : value`).
An object can have properties (attributes) and can also have methods.

An object's properties and methods can be accessed using 2 possible notations:

Dot notation	`person.name`
Bracket notation	`person["name"]`

Dot notation requires that the property name does not begin with a number.



```
var person = {
    name: "Mounder",
    age: 12,
    city: "guelma"
    };
```

1.7 JavaScript language elements

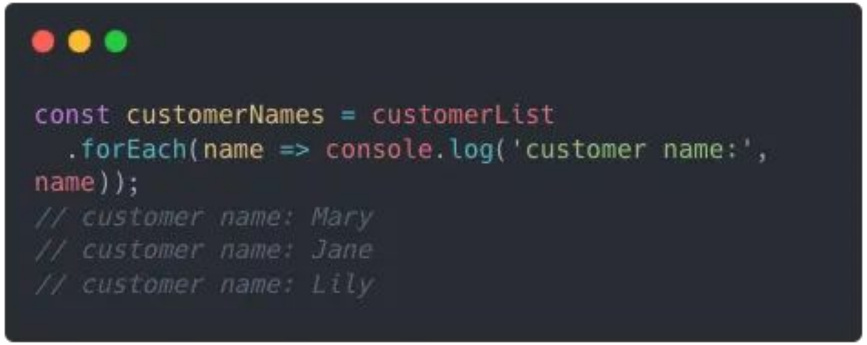
Variables do not have to be declared (using the var keyword) in JavaScript.
But it is advisable to declare them in their functions to avoid certain errors.

➔ Functions on arrays

`sort(); indexOf(); slice(); reverse() lastIndexOf(); splice().`

➔ Iterators

`forEach() ; map()`



```
const customerNames = customerList
   .forEach(name => console.log('customer name:',
name));
// customer name: Mary
// customer name: Jane
// customer name: Lily
```

1.8 JavaScript displays

There are 4 ways of displaying a message in JavaScript:

- ➔ Dialog boxes: (`alert, confirm, prompt`)
- ➔ Writing to the HTML document using `document.write(message)`, which erases all page content and displays the message only.
- ➔ Write to an HTML element using innerHTML(possible with DOM)
- ➔ Write to the browser console using `console.log(message)` which will will display the message in the console and not on the web page.

1.9 Error handling and debugging in JavaScript

JavaScript is a weakly typed language, and is considered one of the most flexible and easiest to learn. However, it has one major drawback: the occurrence of errors.

➔ In the case of syntactic errors, they are relatively easy to correct.

➔ As for logical errors, they are often undetectable and it's up to the programmer to find and correct them. To help the programmer in this difficult task, a debugger for JavaScript is available on the browser (to be seen in Lab).

➔ Verification messages can be displayed in the code not by means of dialog boxes dialogs (`alert, confirm, prompt`) but by `console.log (message)`, which will display the the message in the console and not on the web page (to facilitate verification by the programmer)

2. DOM:Document Object Model

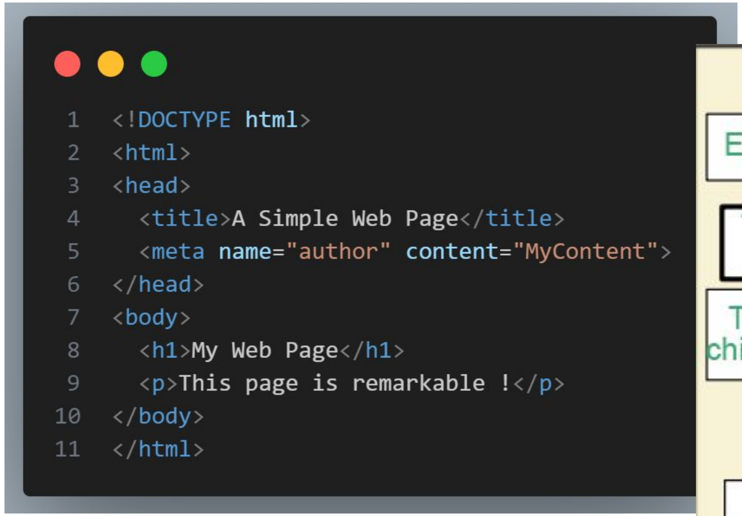
A page loaded into a browser is stored as a tree structure called DOM.

The DOM represents an interface between JavaScript and (HTML+CSS), enabling them to be manipulated by JS programs.

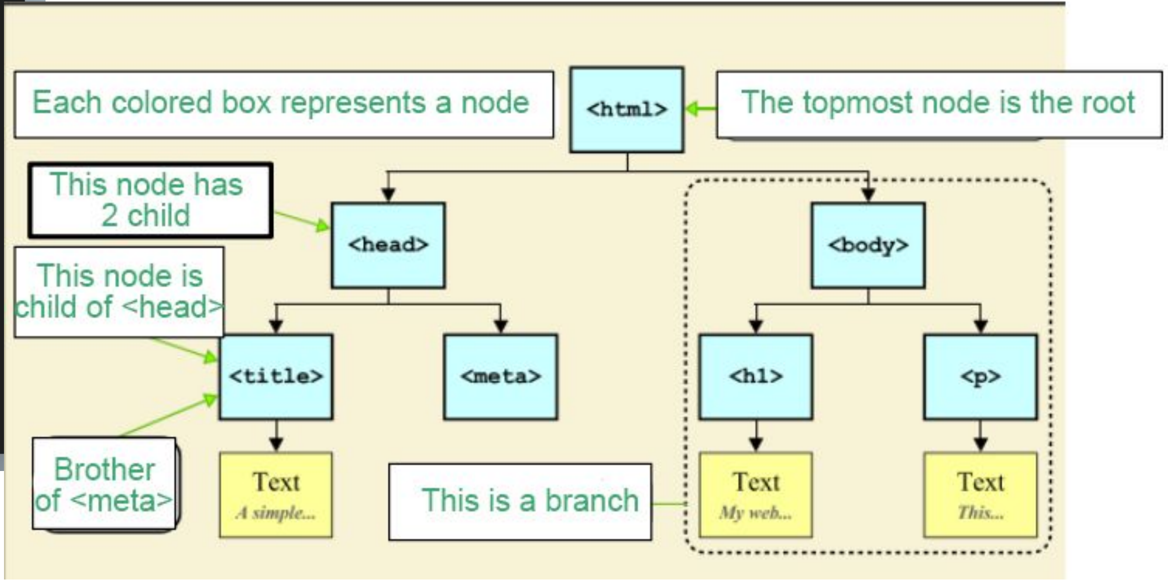
The browser transforms each HTML element into a JavaScript object that can be can manipulate (modify/delete, etc.).

2. DOM:Document Object Model

Example:



```html
<!DOCTYPE html>
<html>
<head>
  <title>A Simple Web Page</title>
  <meta name="author" content="MyContent">
</head>
<body>
  <h1>My Web Page</h1>
  <p>This page is remarkable !</p>
</body>
</html>
```



2.1 Relationships between nodes

The usefulness of the DOM lies in the fact that it allows you to access the various nodes representing the structure of a web page, thanks to the relationships that exist between nodes and the functions used to access them.

In particular, we have the following functions:

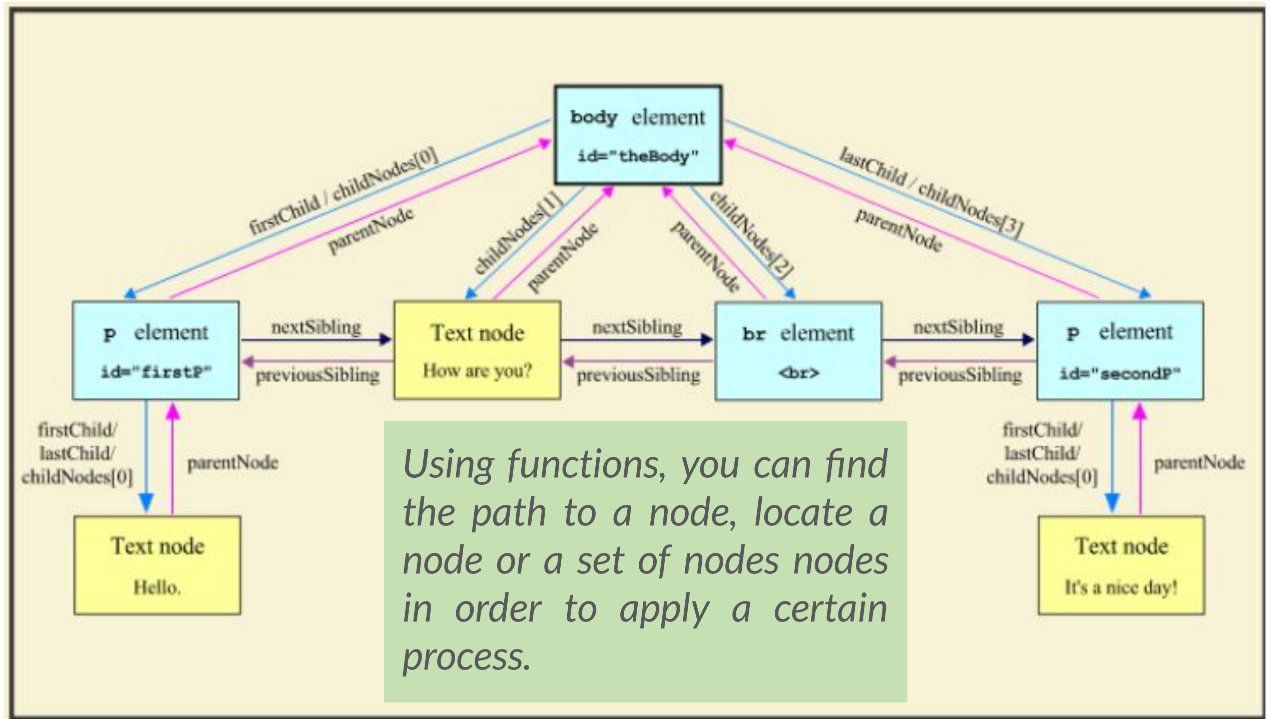
➔ Manipulation of parent node: `parentNode`

➔ Handling child nodes: `childNodes[ ], firstChild, lastChild`

➔ Handling of sibling nodes: `previousSibling, nextSibling`

2.1 Relationships between nodes

Arrangement of nodes and their descendants allows functions to be used as follows:



Using functions, you can find the path to a node, locate a node or a set of nodes nodes in order to apply a certain process.

2.2 Node selection

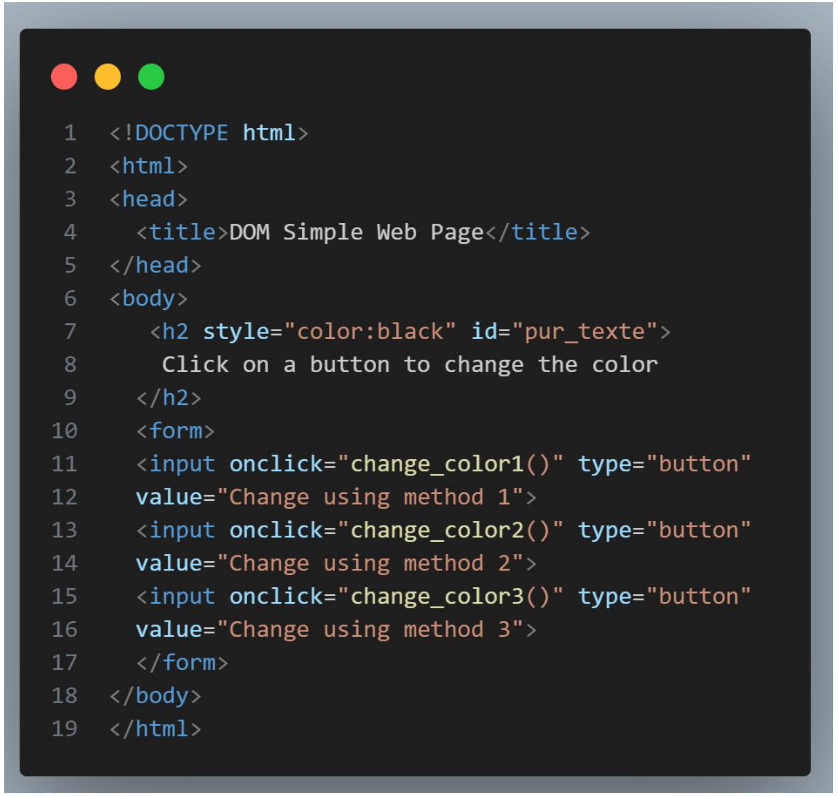
Using the DOM and the preceding functions, you can add, delete, copy or change any node in the DOM. change any node in the DOM. To do this, we use mechanisms for accessing and selecting using one of the following 3 methods:

➔ Method 1: **Use of the exact path:** requires exact knowledge of the there is a risk of error, as the DOM may differ from one browser to another.

➔ Method 2: **Use the element type** (the type of its tag type): `getElementsByTagName( )`, which requires exact knowledge of the tag of the element you're looking for (h2 or h3??), with the risk of several elements having that name.

➔ Method 3: **Using the name of the element itself**: `getElementById( )`. This method is easiest if the element you're looking for has an identifier as follows: `<element_name id="stuff">...</element_name>`

2.2 Node selection



```html
<!DOCTYPE html>
<html>
<head>
  <title>DOM Simple Web Page</title>
</head>
<body>
   <h2 style="color:black" id="pur_texte">
    Click on a button to change the color
  </h2>
  <form>
  <input onclick="change_color1()" type="button"
  value="Change using method 1">
  <input onclick="change_color2()" type="button"
  value="Change using method 2">
  <input onclick="change_color3()" type="button"
  value="Change using method 3">
  </form>
</body>
</html>
```

➔ Method 1:

```
function change_color1(){
document.childNodes[1].childNodes[2]
.childNodes[1].style.color="red";}
```

➔ Method 2:

```
function change_color2(){
document.getElementsByTagName("h2")[0
].style.color="yellow"; }
```

➔ Method 3:

```
function change_color3(){
document.getElementById("pur_texte"
).style.color="blue"; }
```

2.2 DOM selectors

The root of the HTML document is an object called a document, to which a series of methods can be applied to select particular elements of the document.

In particular, we'll be looking at the following methods:

➔ `document.getElementById("id_element")`: returns the element with the value id="id_element"

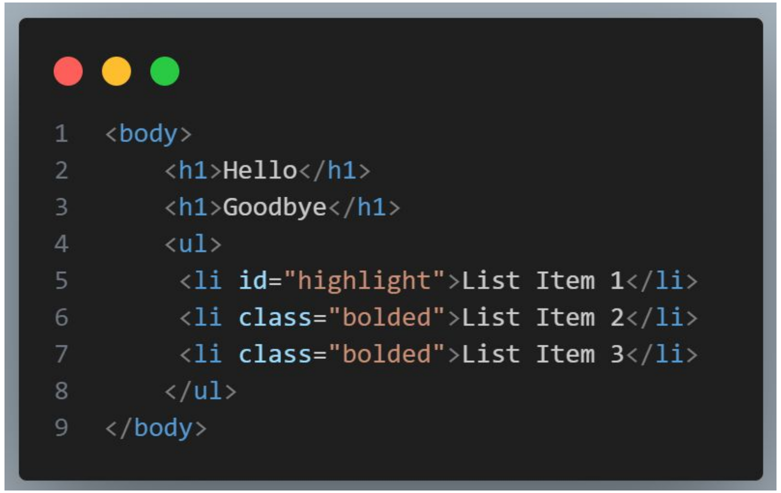
➔ `document.getElementsByClassName("nom_class")`: returns a list of elements with the class class is class_name.

➔ `document.getElementsByTagName("nom_tag")`: returns a list of elements with the tag as tag <tag_name>.

➔ `document.querySelector("selector_CSS")`: can replace all previous methods by specifying a methods by specifying an id, class or tag name in CSS selector notation. It returns the first element satisfying the selecteur_CSS selector

➔ `document.querySelectorAll("selector_CSS")`: works like the previous method but returns an array of all elements satisfying the selecteur_CSS selector, accessible via indices.

2.2 DOM selectors



```html
<body>
    <h1>Hello</h1>
    <h1>Goodbye</h1>
    <ul>
     <li id="highlight">List Item 1</li>
     <li class="bolded">List Item 2</li>
     <li class="bolded">List Item 3</li>
    </ul>
</body>
```

```
var tag = document.getElementById("highlight")
tag = document.querySelector(″#highlight″)
```

It returns: <li id="highlight">List Item 1</li>

```
var tags = document.getElementsByClassName("bolded")
tag = document.querySelectorAll(″.bolded″)
```

It returns: <li class="bolded">List Item 2</li> et <li class="bolded">List Item 3</li>

```
var tags = document.getElementsByTagName("li")
tag = document.querySelectorAll(″li″);
```

It returns:
<li id="highlight">List Item 1</li>
<li class="bolded">List Item 2</li>
<li class="bolded">List Item 3</li>

2.3 DOM-related events

Make pages interactive. There are many events that can be applied to DOM elements: click a button, move to a link, copy and paste, press return, etc.

To support an event, we need to:

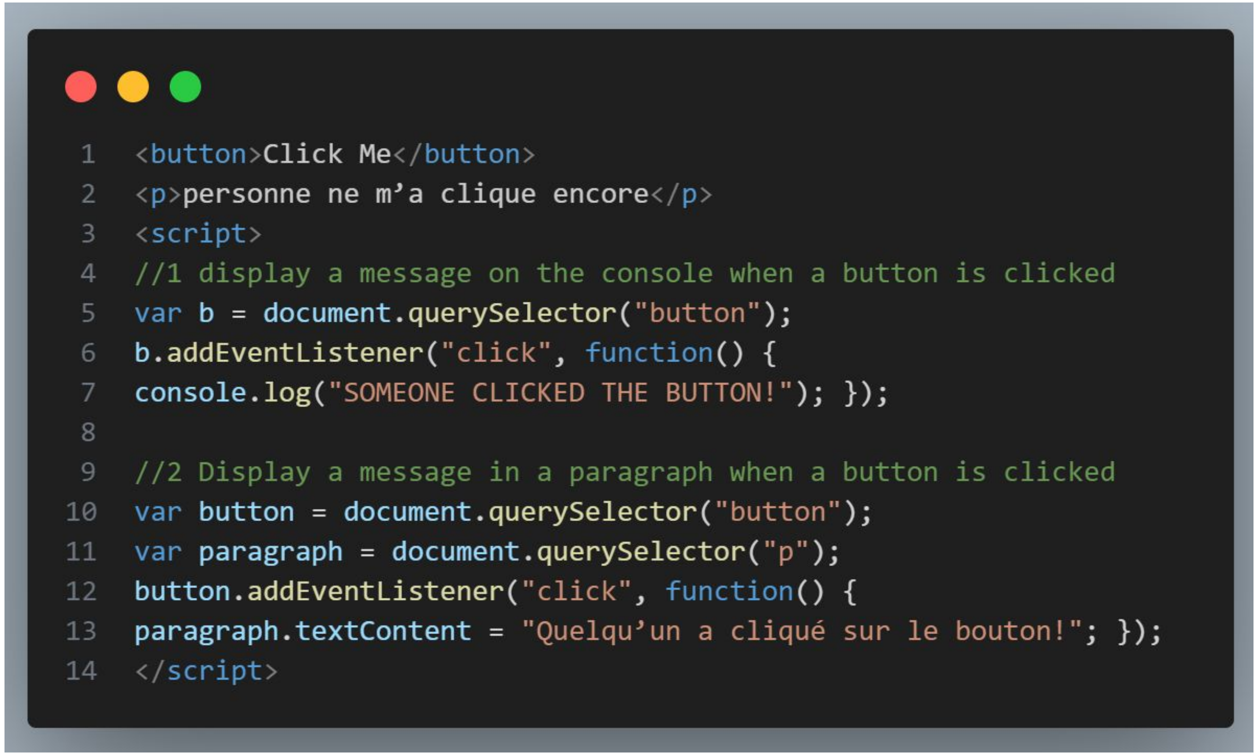
- ➔ Select the element
- ➔ Place an event listener on it, which must perform processing in response to the event.

Syntax:

To add a listener to an element, use a method called addEventListener()

```
element.addEventListener(type, functionToCall());
```

2.3 DOM-related events



```
<button>Click Me</button>
<p>personne ne m’a clique encore</p>
<script>
//1 display a message on the console when a button is clicked
var b = document.querySelector("button");
b.addEventListener("click", function() {
console.log("SOMEONE CLICKED THE BUTTON!"); });

//2 Display a message in a paragraph when a button is clicked
var button = document.querySelector("button");
var paragraph = document.querySelector("p");
button.addEventListener("click", function() {
paragraph.textContent = "Quelqu’un a cliqué sur le bouton!"; });
</script>
```

Lab Exercises Submission Guidelines

➔ **Deadline:**
At the end of each Lab session (no later than Saturday at 23:59)
To: adil.chekati@univ-constantine2.dz

➔ **File's Name to be submitted:**
CAW_Lab%_Gr%_NAMEPair1_NAMEPair2.zip
Example : "CAW_Lab1.part1_Gr1_CHEKATI_BOUZENADA.zip"

Textbook

➔ All academic materials will be available on:

Google Drive.
E-learning platform of Constantine 2 University.
Google Classroom.

SCAN ME!

References

➔ **Book:**
Haverbeke, Marijn - *Eloquent JavaScript: A Modern Introduction to Programming*- (2019)

Online Resource:
Mozilla Developer Network-"JavaScript Guide"
(https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide)

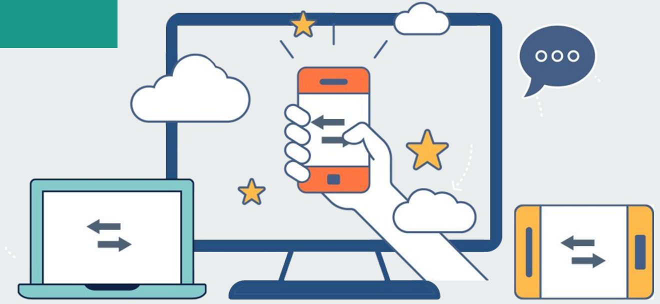
Next Lecture

-Lecture 3-

Chapter 2 – **Advanced JavaScript concepts**

Adil **CHEKATI**, PhD

adil.chekati@univ-constantine2.dz



Questions,
& comments…

📧 adil.chekati@univ-constatine2.dz