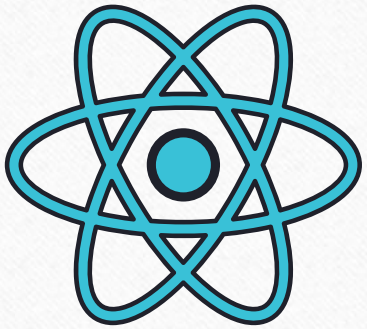
Welcome to JavaScript!



Instructors

Eng. Ahmed Mohamed Abu-Bakr

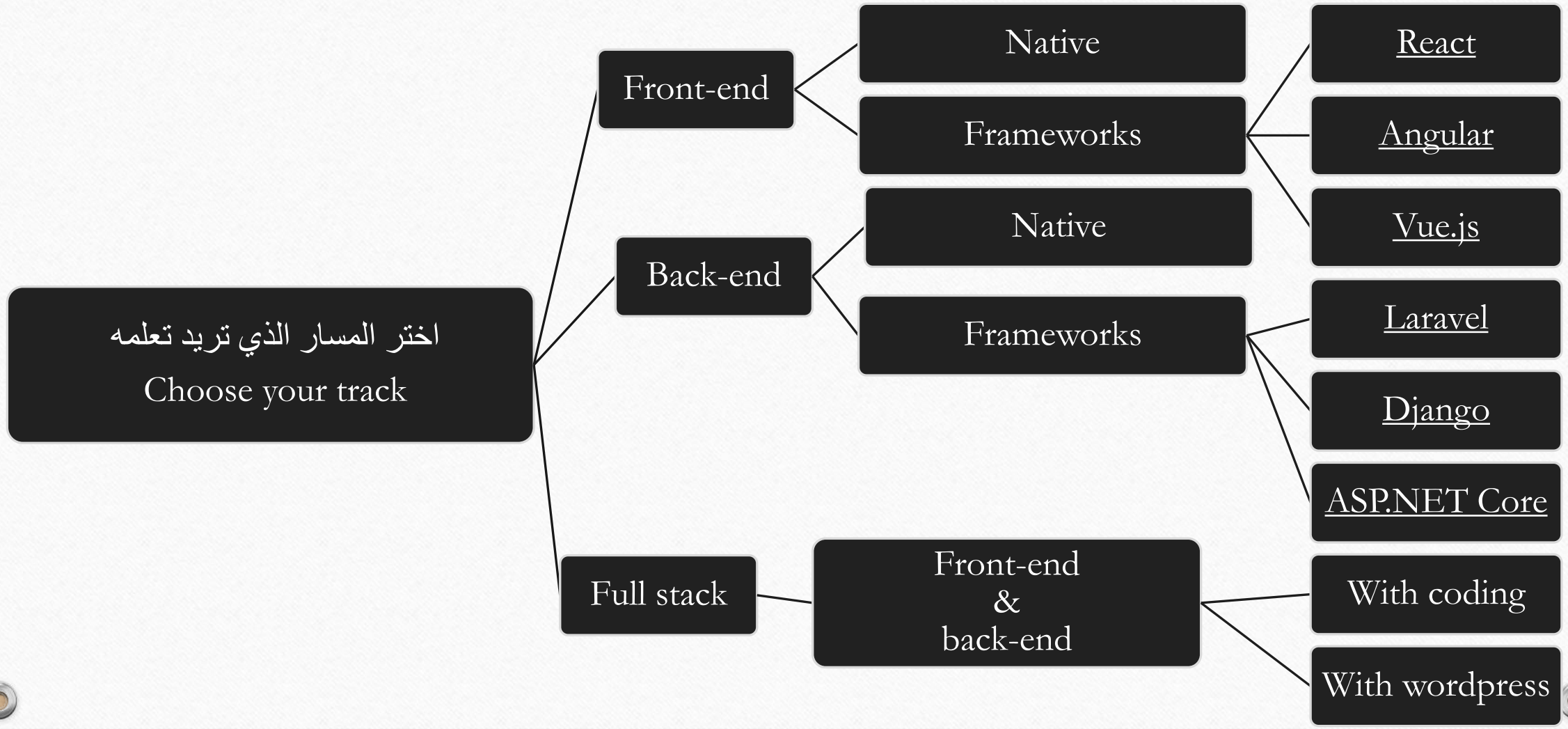
IC Layout engineer
Full stack web developer Laravel-react

Eng. Nada Harby Motawe

Computer science engineer
Full stack web developer


FRONT END
BACK END
comic.browserling.com

كيف تصبح مطور مواقع



Full-stack development roadmaps

خريطة طريق تعلم مسار ال Full-stack development

Front-end development roadmaps

خريطة طريق تعلم مسار ال Front-end development

Back-end development roadmaps

خريطة طريق تعلم مسار ال back-end development

اساسيات بناء مواقع الويب

HTML
بناء الهيكل الاساسي ل مواقع الويب

CSS
اضافة بعض التنسيقات والاحجام ل النصوص والعناوين والصور في المواقع

JS
جعل الموقع حيوي ومتفاعل مع المستخدم

Bootstrap
مكتبة (**Library**) مجاني يستخدم لتسهيل عملية تصميم صفحات الويب على المطور

Advanced
(frameworks + Api + important libraries)

① Syntax to write an HTML element

Opening Tag
Every element has an opening tag with the name of the element at its start.

Attribute and its value (optional)
Attributes are like options of an element. Attributes have value.

```
<name attr="value">
....children
</name>
```

Closing Tag
A closing tag has the name of the element with a forward slash "/" before it.

Children (optional)
Between the opening and closing tags are the children of the element. This can be more elements or just plain text.

```
<name attr="value"/>
```

Self-Closing Tag
Some elements that do not have children do not need a closing tag. In this case a forward-slash "/" is added to the element's opening tag to denote a self-closing tag.

Some examples of self-closing elements are:
img, br, hr, meta, input ....

② Basic markup of every HTML page

```
<!DOCTYPE html>
<html lang="en">
....<head>
........<title>
............page title here
........</title>
....</head>

....<body>
....
....
....  page content
....  goes here
....
....
....</body>
</html>
```

Code Formatting
For good code formatting, remember to indent the children by 2 or 4 spaces.

③ Commonly used HTML elements

```
<h1> ... <h6>
<p>
<i> <strong>
<a>
<div>
<span>

lists:
<ul> <ol> <li>

Special formatting
<blockquote> <pre>

multimedia:
<img /> <video>
<audio>

separators:
<hr /> <br />
```

① Syntax to write CSS

Selectors
The element(s) on which the style should be applied

Property and its value
This is the actual style to be applied to the element(s)

```
selectors {
....property: value;
}
```

② 3 places to write CSS

(A) Inline styles

```
<element style="property: value;">
```

(B) In the <style> element

```
<head>
....<style>
........ selectors { property: value; }
....</style>
</head>
```

(C) In a dedicated file (style.css)
& refer that file via the <link> element

```
<head>
....<link rel="stylesheet"
........ href="style.css" />
</head>
```

③ Selectors and their syntax

Basic Selectors	Combinators		Pseudo Selectors
elementname	selectorA + selectorB	Adjacent sibling	:active
.classname	selectorA ~ selectorB	General sibling	:hover
#idname	parent > child	Direct child	:visited
[attr=value]	parent descendent	Descendent	:focus
*			

④ Common CSS properties (by group)

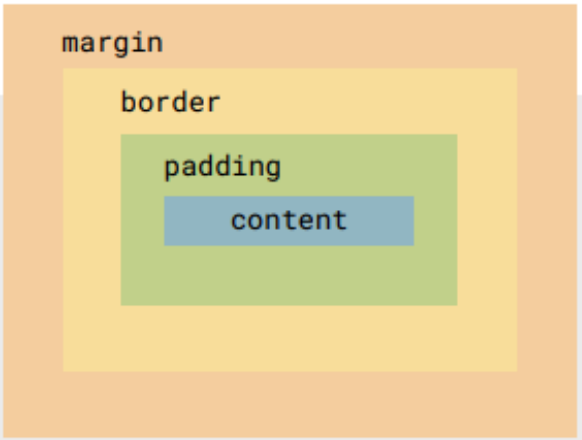
TEXT:
color
font
font-family
font-size
font-weight
letter-spacing
line-height
text-align
text-decoration
text-indent
text-transform
vertical-align

LIST:
list-style
list-style-image
list-style-position
list-style-type

BACKGROUND:
background
background-attachment
background-color
background-image
background-position
background-repeat

DISPLAY:
display
float
clear
overflow
visibility

OTHER:
cursor



BOX:
border
border-color
border-style
border-width
height
margin
padding
width
box-sizing

POSITION:
position
top
bottom
left
right
z-index

اهم عنصر من عناصر الHTML
<form>

امثلة علي العناصر التي تكون بداخل عنصر ال <form>

- <fieldset>
- <legend>
- <label>
- <input>
- <output>
- <textarea>
- <select>
- <datalist>
- <option>
- <optgroup>
- <button>

Input types

1. **text** - creates a single-line text fields(default)
2. **button** - creates a button with no default functionality
3. **checkbox** - creates a checkbox
4. **datetime-local** - creates a date and time picker
5. **email** - creates an input field that allows the user to input a valid email address
6. **file** - creates an input field that lets the user upload a file or multiple files
7. **password** - creates an input field that lets the user enter information securely
8. **submit** - allows user to submit form to the server

اهم عنصر من عناصر الHTML
<form>

تاج ال<form> يحتوي علي مجموعة من المدخلات المختلفة التي تستقبل بيانات من المستخدم بانواع مختلفة ولذلك يعتبر من اهم عناصر الHTML

```
<form action="" method="">
    <label for="firstname">First name: </label>
    <input type="text" name="firstname"  required>
    <br>
    <label for="lastname">Last name: </label>
    <input type="text" name="lastname"  required>
    <br>
    <label for="email">email: </label>
    <input type="email" name="email"  required>
    <br>
    <label for="password">password: </label>
    <input type="password" name="password"  required>
    <br>
    <input type="submit" value="Login!">
</form>
```

"Day 3: JavaScript Fundamentals"

"From Console to Interactive Web"

9:00-9:15 ➤ Intro & Setup

9:15-10:30 ➤ JS Basics

10:30-12:00 ➤ Functions

1:00-2:30 ➤ DOM Magic

2:30-3:30 ➤ Event Listeners

3:30-5:30 ➤ Task Manager Lab

5:30-6:30 ➤ Debugging

"Day 3: What is JavaScript"

JavaScript is a programming language for the web. It adds interactivity to web pages.

```
<button id="sayHy" onclick="alert('Hi')">Click me </button>
```

```
<script>
        document.getElementById("saHay").onclick = function (){
                alert('Hi')
        }
</script>
```

```
//in file-name.js file
document.getElementById("saHay").onclick = function (){
        alert('Hi')
}
```

Type the js code in here

"Day 3: Variables – The Building Blocks"

Storing Data with Variables

```
let name = "Sara";
const PI = 3.14;
```

```
// Different type of data
let quantity = { name: "Ali", age: 22 };
let cart = ‘eggs’ , ‘bread’ , ‘cheese’];
```

let: لما القيمة ممكن تتغير

const: لما القيمة مش هتتغير

var: قديم، بنبعد عنه دلوقتي"

```
var a = "some value";   // functional or global scoped
let b = "some value";   // block scoped
const c = "some value"; // block scoped + cannot get new value
```

"Day 3: Variables & Data Types"

```
let price = 19.99; // Number
const isAdmin = false; // Boolean
let colors = ['red', 'green']; // Array
```

```
// Dynamic type conversion
let quantity = "5";
console.log(quantity + 1); // "51"
console.log(Number(quantity) + 1); // 6
```

① Seven (7) Types

Six Primitive Types

1. String	"Any text"
2. Number	123.45
3. Boolean	true or false
4. Null	null
5. Undefined	undefined
6. Symbol	Symbol('something')
7. Object	{ key: 'value'}
- Array	[1, "text", false]
- Function	function name() { }

```
> var colors = 'red'
<· undefined
> typeof(colors)
<· 'string'
```

⑮ Auto Inherited Properties

When you create a value in JavaScript, certain properties are automatically inherited by this value. This magic happens because every type has a constructor with a special property called prototype. All methods on the prototype gets automatically inherited by the new value created for that type.
Take a look at some of of these methods on the right.

```
const thing = "some text";
```

String

Google 'Mozilla String' to find the docs

```
.concat()
.charAt()
.indexOf()
.startsWith()
.endsWith()
.split()
.slice()
```

```
const num = 123.45;
```

Number

Google 'Mozilla Number' to find the docs

```
.toFixed()
.toPrecision()
.toString()
```

Boolean

Google 'Mozilla Boolean' to find the docs

```
.toString()
```

Array

Google 'Mozilla Array' to find the docs

```
.filter()
.map()
.find()
.every()
.some()
.sort()
.slice()
.splice()
.reduce()
.forEach()
```

⑯ Built-in Objects

Full list of builtin objects in JavaScript visit https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects

JavaScript gives us a ton of useful built-in objects to make our lives easier. The Date and Math objects are very useful on a regular basis. Take a look at some of their features on the right.

Math

Google 'Mozilla Math' to find the docs

```
Math.pow(2, 3)       // 8
Math.sqrt(16)        // 4
Math.min(7, 8, 6)    // 6
Math.max(7, 8, 6)    // 8
Math.floor(123.45)   // 123
Math.ceil(123.45)    // 124
Math.round(123.45)   // 123
Math.random()        // 0.45..
```

Date

Google 'Mozilla Date' to find the docs

```
const d = new Date('9/17/1988');
d.getDay()
d.getFullYear()
d.getMonth()

Date.now()
Milliseconds since Jan 1, 1970
```

"Day 3: Operators & Type Coercion"

Making Calculations & Comparing Values

Arithmetic: +, -, *, / , %

Assign: = , += , =+ , -= , =-

Comparison: ===, ==

```
var x  = 10 + "5" // "105"
Var y  = (10 === "10”) // false
```

Where we use operators do you think?

6 Operators

Full list of JavaScript operators https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators

Operators are reserved-words that perform action on values and variables.

Arithmetic

.. + .. Add
.. - .. Subtract
.. * .. Multiply
.. / .. Divide
.. % .. Remainder
.. ** .. Exponential

Assignment

.. = .. Assign value
.. += .. Add then assign
.. -= .. Subtract then assign
.. *= .. Multiply then assign

Logical

.. || .. Or
.. && .. And

Equality

.. === .. Equality
.. == .. Equality with coercion

Conversion

+ .. Convert to number
- .. Convert to number then negate it
! .. Convert to boolean then inverse it

Relational / Comparison

.. >= .. Greater than or equal to
.. <= .. Less than or equal to
.. != .. Not equal after coercion
.. !== .. Not equal

Increment / Decrement

..++ Postfix increment
..-- Postfix decrement

++.. Prefix increment
--.. Prefix increment

Others

typeof ..
.. instanceof ..
(..)
...spread-operator
.
..[..]
new ..
delete ..
(.. ? .. : ..)

Operator Precedence

Given multiple operators are used in an expression, the "Operator Precedence" determines which operator will be executed first. The higher the precedence, the earlier it will get executed.

Operator Associativity

Given multiple operators have the same precedence, "Associativity" determines in which direction the code will be parsed.

See the **Operator Precedence and Associativity table** here:
http://bit.ly/operatortable

7 Coercion

When trying to compare different "types", the JavaScript engine attempts to convert one type into another so it can compare the two values.

Type coercion priority order:

1. String
2. Number
3. Boolean

```
2 + "7";   // "27"
true - 5   // -4
```

Coercion in action

Does this make sense?

8 Conditional Statements

Conditional statements allow our program to run specific code only if certain conditions are met. For instance, lets say we have a shopping app. We can tell our program to hide the "checkout" button if the shopping cart is empty.

If -else Statement: Run certain code, "if" a condition is met. If the condition is not met, the code in the "else" block is run (if available.)

```
if (a > 0) {
  // run this code
} else if (a < 0) {
  // run this code
} else {
  // run this code
}
```

Ternary Operator: A ternary operator returns the first value if the expression is truthy, or else returns the second value.

```
(expression)? ifTrue: ifFalse;
```

Switch Statement: Takes a single expression, and runs the code of the "case" where the expression matches. The "`break`" keyword is used to end the switch statement.

```
switch (expression) {
  case choice1:
    // run this code
    break;

  case choice1:
    // run this code
    break;

  default:
    // run this code
}
```

9 Truthy / Falsy

There are certain values in JavaScript that return true when coerced into boolean. Such values are called **truthy** values. On the other hand, there are certain values that return false when coerced to boolean. These values are knows as **falsy** values.

Truthy Values	Falsy Values
`true`	`false`
`"text"`	`""`
`72`	`0`
`-72`	`-0`
`Infinity`	`NaN`
`-Infinity`	`null`
`{}`	`undefined`
`[]`	

"Day 3: Functions Deep Dive"

Two famous types of writing functions

```
// Function Declaration
function calculateTax(price) {
    return price * 0.08;
}

// Arrow Function
const calculateTax = (price) => price * 0.08;
```

Common Mistake:

```
// Missing return
function double(num) {
    num * 2; // Oops!
}
```

Never forget :-

Input → process → output

Fast excersize create function greets any one enter the site

Hint to take value from him use prompt()

"Day 3: DOM Manipulation"

DOM is short for : Document Object Model

```
// Access Element
const myDiv = document.getElementById('theDiv');
newDiv.textContent = "Hello!";

// Modify style
newDiv.style.backgroundColor = "yellow";
```

Exercise:

1- Change element text and color in your by it's id

2- Change all <h2> colors to blue. 😈

"Day 3: DOM Manipulation"

DOM is short for : Document Object Model

```
var parentDiv = document.getElementById('theParent');

// Access Element
var myDiv = document.getElementById('theDiv');
var theCopy = myDiv.cloneNode(true);

// Output The Duplicated
parentDiv.appendChild(theCopy);
```

Exercise:

make the duplicator and make him ask for name. 😈

Team Members

John Smith

Lead Web Developer

Specialized in web application development using modern technologies like React and Node.js. Over 5 years of experience in the field.

HTML CSS JavaScript React

View Profile Contact

Sarah Johnson

UI/UX Designer

Specialized in user experience and interface design. Works on improving user experience and making applications more user-friendly.

Figma UI Design UX Research Prototyping

View Profile Contact

Michael Brown

Mobile App Developer

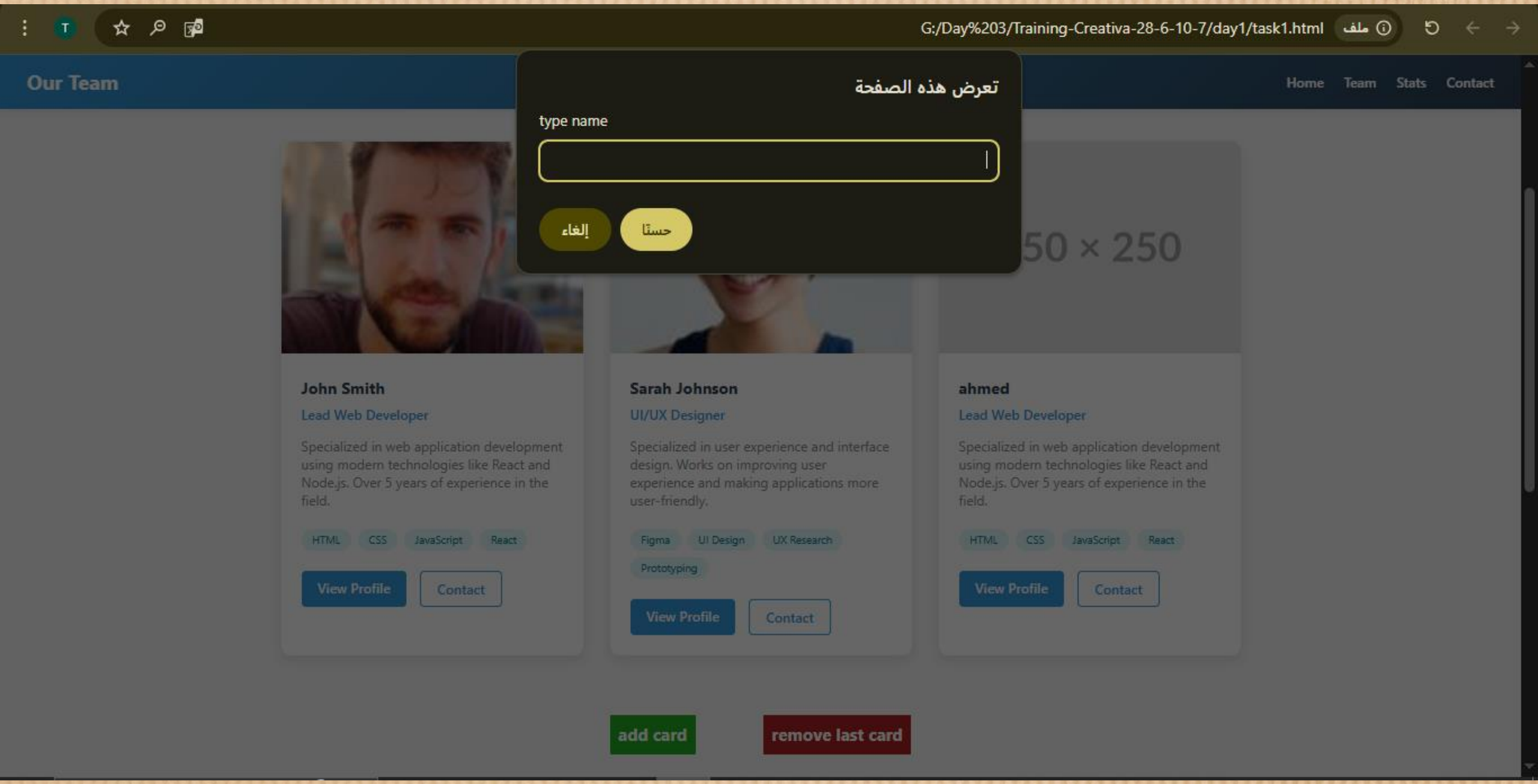
Expert in mobile app development using React Native and Flutter. Focuses on building high-performance, user-friendly applications.

React Native Flutter Firebase Mobile UI

View Profile Contact

add card

remove last card


G:/Day%203/Training-Creativa-28-6-10-7/day1/task1.html
ملف
Our Team
Home
Team
Stats
Contact
تعرض هذه الصفحة
type name
إلغاء
حسنًا
50 × 250
John Smith
Lead Web Developer
Specialized in web application development using modern technologies like React and Node.js. Over 5 years of experience in the field.
HTML
CSS
JavaScript
React
View Profile
Contact
Sarah Johnson
UI/UX Designer
Specialized in user experience and interface design. Works on improving user experience and making applications more user-friendly.
Figma
UI Design
UX Research
Prototyping
View Profile
Contact
ahmed
Lead Web Developer
Specialized in web application development using modern technologies like React and Node.js. Over 5 years of experience in the field.
HTML
CSS
JavaScript
React
View Profile
Contact
add card
remove last card

John Smith

Lead Web Developer

Specialized in web application development using modern technologies like React and Node.js. Over 5 years of experience in the field.

HTML CSS JavaScript React

View Profile Contact

Sarah Johnson

UI/UX Designer

Specialized in user experience and interface design. Works on improving user experience and making applications more user-friendly.

Figma UI Design UX Research Prototyping

View Profile Contact

ahmed

Lead Web Developer

Specialized in web application development using modern technologies like React and Node.js. Over 5 years of experience in the field.

HTML CSS JavaScript React

View Profile Contact

add card

remove last card

⑭ DOM - Document Object Model

Query/Get Elements

```
// Preferred way:
document.querySelector('css-selectors')
document.querySelectorAll('css-selectors', ...)

// Old ways, and still work:
document.getElementsByTagName('element-name')
document.getElementsByClassName('class-name')
document.getElementById('id')
```

Create / clone Element

```
document.createElement('div')
document.createTextNode('some text here')
node.cloneNode()
node.textContent = 'some text here'
```

Add node to document

```
parentNode.appendChild(nodeToAdd)
parentNode.insertBefore(nodeToAdd, childNode)
```

Get Element Details

```
node.nextSibling
node.firstChild
node.lastChild
node.parentNode
node.childNodes
node.children
```

Modify Element

```
node.style.color = 'red'
node.style.padding = '10px',
node.style.fontSize = '200%'

node.setAttribute('attr-name', 'attr-value')
node.removeAttribute('attr-name')
```

Get and Modify Element Class

```
node.classList
node.classList.add('class-name', ...)
node.classList.remove('class-name', ...)
node.classList.toggle('class-name')
node.classList.contains('class-name')
node.classList.replace('old', 'new')
```

Remove Node

```
parentNode.removeChild(nodeToRemove)
// Hack to remove self
nodeToRemove.parentNode.removeChild(nodeToRemove)
```

Events

```
node.addEventListener('event-name', callback-function)
node.removeEventListener('event-name', callback-function)
```

List of Events: https://developer.mozilla.org/en-US/docs/Web/Events
or google *"Mozilla event reference"*

What is a "Node"? (in the context of DOM)

Node: Every item in the DOM tree is called a node. There are two types of node - A text node, and an element node:

Text Node: Node that has text.

Element Node: Node that has an element.

Child Node: A node which is a child of another node.

Parent Node: A node which has one or more child.

Descendent Node: A node which is nested deep in the tree.

Sibling Node: A node that share the same parent node.

"Day 3: Warmup Exercise"

"From Console to Interactive Web"

```
// Before (Buggy)
document.querySelector('button').addEventListner('clik', myFunction);
```

```
// After (Fixed)
document.querySelector('button').addEventListener('click', myFunction);
```

Discussion Point:
"Why do we need exact syntax in programming?"

"Day 3: Event Listeners"

```
<button id="greetMe"> say hi to me </button>
```

```
// Active greeting message
const greetButtun = document.querySelector('greetMe');
greetButtun.addEventListener('click', () => {
        // here take the value from user
        if (name == "ahmed") {
                alert("hi admin");
        }else {
                alert("hello" + name)
        }
});
```

```
// Active greeting message
const greetButtun = document.querySelector('greetMe');
greetButtun.onclick('click', (name) => {
        if (name == "ahmed") {
                alert("hi admin");
        }else {
                alert("hello" + name)
        }
});
```

Exercise:-
Add your event listener

"Day 3: Event Listeners"

```
// Form validation
const form = document.querySelector('form’);
form.addEventListener('submit', (e) => {
        if (!validateInput()) {
                e.preventDefault();
                showError("Fix errors before submitting!");
        }
});
```

Exercise At home:-

Create a login page and save user’s name also use it in another page

"Day 3: Task Manager Lab"

Milestones:

1. Basic add/display (1 hour)
2. Add delete buttons (30 mins)
3. Persist to localStorage (Bonus)

```
const taskInput = document.getElementById('taskInput’);
const addBtn = document.getElementById('addBtn’);

addBtn.addEventListener('click', () => {
        if (taskInput.value.trim() !== ' ') {
                addTask(taskInput.value);
                taskInput.value = ' ';
        }
});
```

"Day 3: Debugging Session"

```
// Buggy Code
function loadTasks() {
    const tasks = localStorage.getItem('tasks’);
    return tasks;
}

// Fixed Version
 function loadTasks() {
    return JSON.parse(localStorage.getItem('tasks')) || [];
}
```

"Day 3: Storing element"

- localStorage:
  تخزين دائم حتى الحذف يدويًا حجم كبير حتى (5-10MB) لا تُرسل للسيرفر

- sessionStorage:
  يُمسح عند غلق التبويب مشابه لـ localStorage لكن مؤقت

- Cookies:
  تُرسل مع كل طلب للسيرفر حجم صغير (4KB) يمكن تحديد مدة الانتهاء

"Day 3: Basic Operations with localStorage"

- إضافة أو تعديل: setItem(key, value)
- قراءة: getItem(key)
- حذف قيمة: removeItem(key)
- مسح الكل: clear()

مثال:

```
localStorage.setItem("color", "blue");
let favColor = localStorage.getItem("color");
localStorage.removeItem("color");
localStorage.clear();
```

"Day 3: Practical Example: Save User Name"

- نستخدم input لحفظ اسم المستخدم في localStorage.
-نعرض الاسم مرة تانية لما المستخدم يرجع.

مثال:

```
function saveName() {
  let name =
document.getElementById("nameInput").value;
  localStorage.setItem("userName", name);
  document.getElementById("output").textContent =
"Welcome " + name;
}
```

```
function saveName() {
  let name =
document.getElementById("nameInput").value;
  localStorage.setItem("userName", name);
  document.getElementById("output").textContent =
"Welcome " + localStorage.getItem("userName");
}
```

The End

in the end I hope you understood all I said contact on :

https://www.facebook.com/abobakr143

https://wa.me/201113284597