JavaScript Course Jonas → udemy 70 hours

Date ...../...../..... No ....

* كتابة كود في ملف HTML عن طريق <script>

* لربط ملف script.js في HTML

<script defer src="—">

console.log ((Variable Name)); → للطباعة في console

let NameOfVariable = "القيمة" ; → when you change the value you should write let

Variables
- Camel Case is the common for js in writing variables
- Not start with number
- Not contain $
- You shouldn't start variable name with uppercase letter like Person → it should be person
  - this, we will use it in OOP
- it must be understant when you read the value of it to be a clean code

Primitive data type :-

1- Number
2- String
3- boolean
4- undefined → with empty value → let year
5- Null → " " "
6- Symbol (ES 2015)
7- BigInt (ES2020)

data type in js automatically determined, so you don't need to write data type in js (Not other languages)

let	Const	Var
قيم ممكن نغيرها عادي	قيم ثابتة لا تتغير أبداً	old way to define variables
let age = 30 ; age = 32 ;	Const birthday = 1991 ;	Var job = 'Programer' ; job = 'teacher' ;

MDN Documentation

(Window + . → write emoji)

Operators

Date ...../...../..... No ....

Math operator → + , - , * , / !مع المتغير او الارقام اللي بعمل عليها

Assignment operator → x += 10 , x++
x *= 4 , x--

Comparison operator → > , < , <= , >=

Operator precedence :-
* الترتيب هو ترتيب العمليات اللي بيمشي في المعادلة نفسها .

Strings and Template Literals
↓ it start with `` → backtick characters
↓ it allow multiple lines
↳ ${ any values } → ${ — } instead of (+)
↓ it is best and clean another (+)

if Condition , else
if ( ) {
:
} else if
... }
} else

if → else if → else

Type Conversion and Coercion : Converting from type to another
↓ You make it manually
↓ it made automatically

Const Year = '1991';
Console.log ('Input Year' + 18) = 199118
Console.log (Number(Year) + 18 = 2009
Console.log (Number ('Jonas') ; → NAN
↓ Not A Number

Const money = 0 ;
if (money) {
Console.log ('Don't Spend it');
} else {
Console.log (' get job');
}

When money = 0 → false → get a job
money ≠ صفر → Don't Spend it

Dar Kareem

Date ....../....../...... No ....

5 falsy Values → 0 , '' , undefined , null , NAN

Strict

===

'23' !== 23

↓

for Clean Code

Loose

==

'23' == 23 ✓

it has Coercion

→ will make many bugs with you

Switch Case

```
Switch (   ) {
  Case '   ' :
    ====
    break ;
  Case '   ' :
    ====
    break
  default :
    Console.log('Not Valid Value') ;
}
```

Conditional operator [Ternary]

الشرط ? ناتج الشرط : الناتج المخالف للشرط ;

مثال على if و else مختصر

age >= 18 ? 'adult' : 'boy' ;

'use strict' ; → لو كتبت متغير وجيت كتبته بعد كده نسيت حرف فلاقيته هيعتبر انه هو تعشي هيطلعلك error

function اسم (Parameter1 , Parameter2 ......) { → Creating the function
  ====
  ====
}

اسم (argument , ,) ; → Calling // invoking // running the function

you can store it in a variable → named **function expression**

Dar Kareem

method is a function which attach to object.

Arrow Function

Date ..../..../.... No ....

Conts متغير = Parameter => ما تعمله الفانكشن ;

Const Age = birthYear => 2037 - birthYear;
const age1 = Age(1991);

data Structure

[1] Arrays → const اسم = [, , ----];
~ ~ = new Array(, , ----);

(methods)

اسم المتغير.push();

add element at the end of Array → Push | remove Pop
~ ~ ~ ~ Start ~ ~ → unShift | Shift

object

const اسم = {

}

(operators on object)

Console.log(اسم . العنصر); → Dot Notation
Console.log(اسم ['العنصر']) → Bracket Notation

→ give you the correct answer

(for loop)

for (let i = 1 ; i <= 10; i++){
}

while (i <= 10) {
i++;
}

Math.random → 0 : 1

1 + (Math.random * 100) → 1 : 100

Date ... / ... / ... No ...

a + Math.random * (b - a) → random values from (a to b)

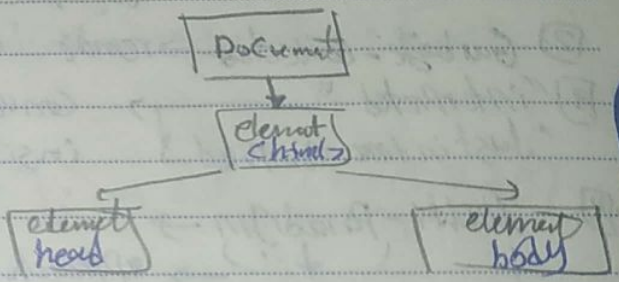
Math.trunc ( ) → convert from ~~str~~ double to integer

Section 5 Advices → You should have a goal →

* Care about the code how it works
* make challenges and Practise coding
imagine big project you want to make it

Section 7 Guess My Number

DOM : Document Object Model :- structured representation of HTML documents, Allows JS to access HTML elements and style to manipulate them.



Guess My Number ↓ for only selector

Date ... / ... / ... No ...

Refactoring our code → the dry principle
→ try Not to duplicate the code

after finish of the Project You should refactor it - by decreasing the code lines

* How Javascript Works behind the scenes

Definition Javascript is :-

1. High-level → every thing happens automatically Not like C language You should create the variable and give it a type in JS You don't need to do that
2. Garbage-Collected → remove old unused object automatically.
3. "interpreted" / "Just in time compiled" → Conversion to 0, 1 "Machine Code" inside the JS engine
4. Multi-Paradigm → (+) → approach and mindset of structuring code.
   - 1- Procedural Programming
   - 2- OOP
   - 3- Functional Programming FP
5. Prototype-based object oriented
6. First class Functions
7. Dynamic
8. Single-threaded
9. Non-blocking event loop

Dar Kareem

Section 9

Date ... / ... / ... No ...

Destructuring Arrays

Also

```
Const colorArr = ["red", "yellow", "blue", "green"];
Const [first, second] = colorArr;
Console.log(First, Second);
```

temp = x;
x = y;
y = temp;

You can swap elements by it instead of temp method

```
let [Pos1, Pos2] = colorArr;
[Pos1, Pos2] = [Pos2, Pos1];
```

Also

Destructing objects

```
Const frontEnd = { frontend: "React", backend: "Node", database: "MongoD
Const { frontend, backend } = frontEnd;
Console.log(frontend, backend);
```

Spread Syntax ...

لتوسيع كل عناصر المصفوفة بدلاً من تكرارها

```
Const arr = [7,8,9]
Const newArr = [1,2,3, ...arr];
```

النتيجة [1,2,3,7,8,9]

for copy array

```
const x = [...arr1, ...arr2];
```

Also

Reset Pattern and Parameters

```
Const add = function (...numbers) {
    let sum = 0;
    for (let i=0; i< numbers.length; i++)
    sum += number[i];
    Console.log(sum);
}
add(2,3,4,5);
add(8,2,5,10,12);
add(1,5,3);
```

متى ما حدد يعد parameters

Date ... / ... / ... No ...

Nullish operator (??) → null and undefined (NOT 0 or '')

for of loop

for (const el of array) console.log(el);

optional Chaining (?.) looping objects

data structure

1) Sets const s = new Set([1, 1, 2, 3]);

- s.size → 3
- s.has(1) → true, false
- s.add(4)
- s.delete(3)

there is No index here

2) Maps → map values to key

const m = new Map()

m.set(key, value)

const rest = new Map();

rest.set('name', 'Mohamed');

rest.set(1, 'ok');

rest.get('name') → Mohamed

.has(_);

.delete('name');

const m = new Map([
  [key, value],
  [,],
  [,],
]);

| Working with Strings |

Date ...../...../..... No ....

[Ex]

```
Const Plane = 'A320';
Console.log ( Plane [0] ); → A          وهكذا Plane[1] → 3
             Plane.Length → 4
             'Moh'.Length → 3
             'Moh' [0] → M
             Plane.indexOf ('3') → 1
             ~ .slice (5)        → الـ index بيبدأ من بعدها الكلام
             .slice (4,7);       "هنا" البداية والنهاية للـ string الناتج
             lastIndexOf ( )
             .toLowerCase ( );
             toUpperCase ( );
       £ → $ → replace ('$', '£');
             replace (/$/g, '£');
```

علشان نغير كل الموجود في الـ Paragraph مثلاً

```
Console.log ('a + very + nice + string'.split ('+'));   → "a", "very", "nice" ~ string
Const [FirstName, lastName] = 'Mohamed Said'.split(' ');
                                 ← يفرق بينهم بعد المسافة
const Me = ['Mr.', firstName, lastName.toUpperCase()].join (' ');
                  → Mr. Mohamed SAId
                  Mr.++ Mohamed++ SAId ← join ('++');
```

We use them in CreditCard
****** 433 مثلا

```
[ PadStart (20, '+') → هيزود في أول الكلام +++ لحد ما يكون عدد الـ string ← 20
[ PadEnd   (20, '*') → هيزود من الخلف
```

56

Pass by Value

عشان تتطلع

Variables

it made by copy it

if you change Variable in function this Value will be only in the function but out of it, Value will be the main Value of it

Pass by refrense / by sharing

function, methods / objects

→ by reference so if it changed in function it will be the same out of it

bind method

Immediatly

Immediately Invoked Function Expression [IIFE]

↳ it mean making function without name and calling it

ex

```
(function () { console.log("Moh") }) () ;   → Moh

(() => console.log("Moh")) () ;
```

closures

```
Const g = function () {
    Const a = 23 ;
    f = function () {
        Console.log (a * 2) ;
    }
}
```

عن دالة جوا دالة تانية ومتعرفها المتغير اللي في الدالة الأم

Array Methods

Date ...../...../..... No ....

slice Method

```
            0    1    2    3    4
let arr = ['a', 'b', 'c', 'd', 'e'];
console.log( arr.slice(2) );   →  c, d, e
                .slice(2, 4);  →  c, d
                 slice(-2)     →  d, e
arr.slice() = [...arr] = كل العناصر
```

Splice Method

↳ It extract from the orginal array so it delete any items of the orginal array.

```
arr.splice(2) → c, d, e   after → arr = ['a', 'b']
```

reverse Method

```
arr.reverse() → هيقلب arr
```

هتشتغل على العنصر وبعدها متغيرة بشكل دائم

Concat

```
arr1.concat(arr2)
```

لم مصفوفتين مع بعض

join

```
arr.join(' ')
```

علامة ما هتكتبه جوا هيكون هو الفاصل بين عناصر المصفوفة

at Method

```
arr[0] = arr.at(0);
arr[arr.length] = arr.at(-1);
```

هنستخدمها لما نوصل لآخر عنصر بسهولة

Looping Array

For each

```
arr.forEach(function (el) {
   
});
```

عنصر الأريب اللي (element, index, array) → (el, i, arr)

for of

arr.entries → [index, ele] → [i, ele]

```
for (const el of arr) {

}
```

Data Transformation

Date ..../..../.... No ....

Map	Filter	Reduce
Ex: current * 2	current > 2	acc + current
make new array with new values, result to the operation you want	returns new array containing only elements which pass the test condition	make all array elements on one value

ملاحظة

- Maximum value → [illegible] for loop

```
const max = movements.reduce((acc, mov) => {
    if (acc > mov) return acc;
    else return mov; }, movements[0]);

console.log(max);
```

← [illegible]

[illegible]

Chaining Methods

Math.abs( ) → | | → [illegible]

find method

find method → return the first element of the array that satisfy the condition.

```
find(mov => mov < 0)
```

Scanned by CamScanner

Date ..../..../.... No ....

const arr = [[1,2,3], [4,5,6], 7,8] ; Flatten method
arr.flat( ); → (1,2,3,4,5,6,7,8)

Section 12

Number ('23') ; + (' ') ;

Number.ParseInt ('30px') → 30
~ ~ ~ ('e23') → NaN → Not a Number
~ . ParseFloat ('2.5px') → 2.5
~ . isNaN (20) → false (لو حاجة Not NaN ← هيطلع true)
~ . isFinite (20) → true
~ . isInteger (23/10) → false

Math.Sqrt (25) → = 25 ** 1/2 أس
Math.Max (حط أرقام)
~ . min (- - -)
~ . random()
~ . round() → التقريب العادي بالأكبر من 5، والأصغر ينزل للأقل
~ . ceil() → تقريب للأكبر
~ . floor() → للأصغر
~ . trunc () → يشيل الكسور ويرجع صحيح (ولا يقرب ولا يبعد)
+(2.345) toFixed (2) → 2.35 تقريب الكسر نفسه
بيرجع string

التعامل مع أرقام كبيرة
BigInt ( ) = '345671231234567' n
$2^{53}-1$ = هو آخر رقم تقدر تكتبه صحيح

Dates & time
const ↳ new Date( )
const future = new Date (2037, 10, 19, 15, 23);
console.log (future.getFullYear()) → 2037
~ . getMonth() → 10
getDate () → 19
hours
Dar Kareem
get second ( );

Date.now ()

Date / / No

Operation with dates

you should transform date to Number ( ) then make Operation on it

Ex:- const calcDaysPassed = (date1, date2) =>
Math.abs(date2 - date1) / (1000 * 60 * 60 * 24);
const days = calcDaysPassed(new Date(2037, 3, 4),
new Date(2037, 3, 14);

Internationalizing Dates Intl

= new Intl.DateTimeFormat('ar-SY').format(now); → ٢٠٢٠/١٢/٨

(en-US) الولايات المتحدة → 8/12/2020

ISO language Code Table

new Intl.NumberFormat(navigator.language, options);

style: 'currency', unit: 'celsius'

CTRL + SPACE

Date ... / ... / ... No ...

Timers

ms

setTimeout (( ) => { }, 3000);

setInterval (( ) => { }, 1000);

OOP DOM

- allows us to make JavaScript interact with the browser
- DOM tree is generated from an HTML document
- DOM is very complex API : Application Programming Interface
  - that contain lots of methods and properties to interact with DOM tree → querySelector() / addEventListner()

Selecting elements in JS

```
document.querySelector('.header');
        .querySelectorAll('.section');
        .getElementById('menu');
        .getElementsByTagName();
        .getElementsByClassName('btn');
```

Creating and inserting elements

1. .insertAdjacentHTML
2.

```
const message = document.createElement('div');
message.classList.add('cookie-message');
message.innerHTML = 'We use cookies for improved functionality and analytics. <button class="btn">Got it!</button>';
header.append(message);
header.prepend(message);
header.before(message);
header.after(message);
```

Delete elements

```
message.remove();
```

Dar Kareem

js

Styles

Date ... / ... / ... No ...

— . style . backgroundColor = '#333' ;
, — . width = '120%' ;
, — . height = Number.parseFloat(getComputedStyle(—).height
, 10) + 30 + 'px' ;

document . documentElement . style . setProperty ('class-name', 'value')
('--color-primary', 'orange')

Attributes

— . alt = 'value' ;
— . setAttribute ('name', 'value')

Classes

— . classList . add ('', '')
. remove ('')
. toggle ('')
. contains ('')

OOP → are linked to Prototype object

Date ... / ... / ... No ...

based on the concept of object → solution of Spaghetti code to make it flexible and easy to maintain

4 fundmetal OOP Principles

- Abstraction
  - abstract class has name of function and variables without values
- Encapsulation
  - Properties → Private not accessible from out of the class
- Inheritance
  - child extend Parent
- Polymorphism
  - child class can overwrite the method which is inherited from Parent

Constructor function

1) new
2) function is called, this. ——
3) {} linked to Prototype
4) function automatically return {}

```
const Person = func (...
  this.1 = 1
  this.2 = 2
}
const = new Person (✓, ✓)
```

↓ object created

You should never write function on constructor because a lot of People will use it in the log in for example so the Performance will be little

↓ Solution for that we use Prototype

```
Person.Prototype.calcAge = function () {
}
```

then call it normal

```
const Jack = new Person (——, ——)
Console.log (Jack.__proto__);  → Parent object
Jack.__proto__ === Person.Prototype → true
Person.Prototype.isPrototypeOf (Jack) → true
Person ~ ~ ~ ~ (Person) → false
```

Dar Kareem

Date ... / ... / ... No ...

* Prototype is come from [new] → when you make new object
  With [new] it inherit it
  [also] → when you make any → arr = [3, 6, 7, 9]
  it inherit all the method by Prototype
  console.log(arr.__proto__) → [methods of Array]
  [methods of Proto]

ES6 classes

Class expression

```
const PersonCl = class {

}
```

Class declaration

```
class PersonCl {

}
```

```
class Person {
  constructor (firstName, birthday) {
    this.firstName = firstName;
    this.birthYear = birthYear;
  }
  * method will be added to Prototype property.
  calcAge () {
    console.log( 2037 - this.birthYear );
  }
}

const Mohand = new Person ('Mohand', '2001');
console.log (Mohand);
```

Classes are Not hoisted, are first-class citizes
are executed in Strict mode.

Getter , Setter

Password → allah See You

Date ....../....../...... No ....

```
get age () {
    return 2037 - this.birthYear ;
}

set fullName(name) {

}
```

→ console.log(Mohamed . age);

→ 2037 - 2001 =

→ Mohamed . fullName ('Mohamed')

```
const Mohamed
= Object.create ( PersonProto )
```

Create object Without using new operator

```
Class StudentCl extend PersonCl {
  constructor ( fullName, birthYear, course) {
  // → always needs to happen first.
  super ( fullName, birthYear ) ;
  this.course = course ;
  }
```

عنصر → Private

```
Static method - {
}
```

For Performance → ASYNC → defer → head
script js
Date / / No

Never Put script of JS in HTML → head Without → defer → ASYNC
always Put them in the End of body

Section 16

Synchronous

* Code is executed line by line
* each line Waits for previous line to finish
* long running operation block code execution.

Asynchronous

* Ex: timer → setTimeout (func — 3000)
* executed after a task that runs in background finishes.
* Non blocking
* Call back function doesn't make code asynchronous

AJAX Asynchronous JavaScript And Xml : allows us to communicate With remote web Servers in an asynchronous Way.
+ With AJAX calls, We can request data from web Servers dynamically

API

Application Programming Interface : Piece of Software that can be used by another Piece of Software, in order to allow applications to talk to each other.

types :- DOM API, Geolocation API, own Class API, online API

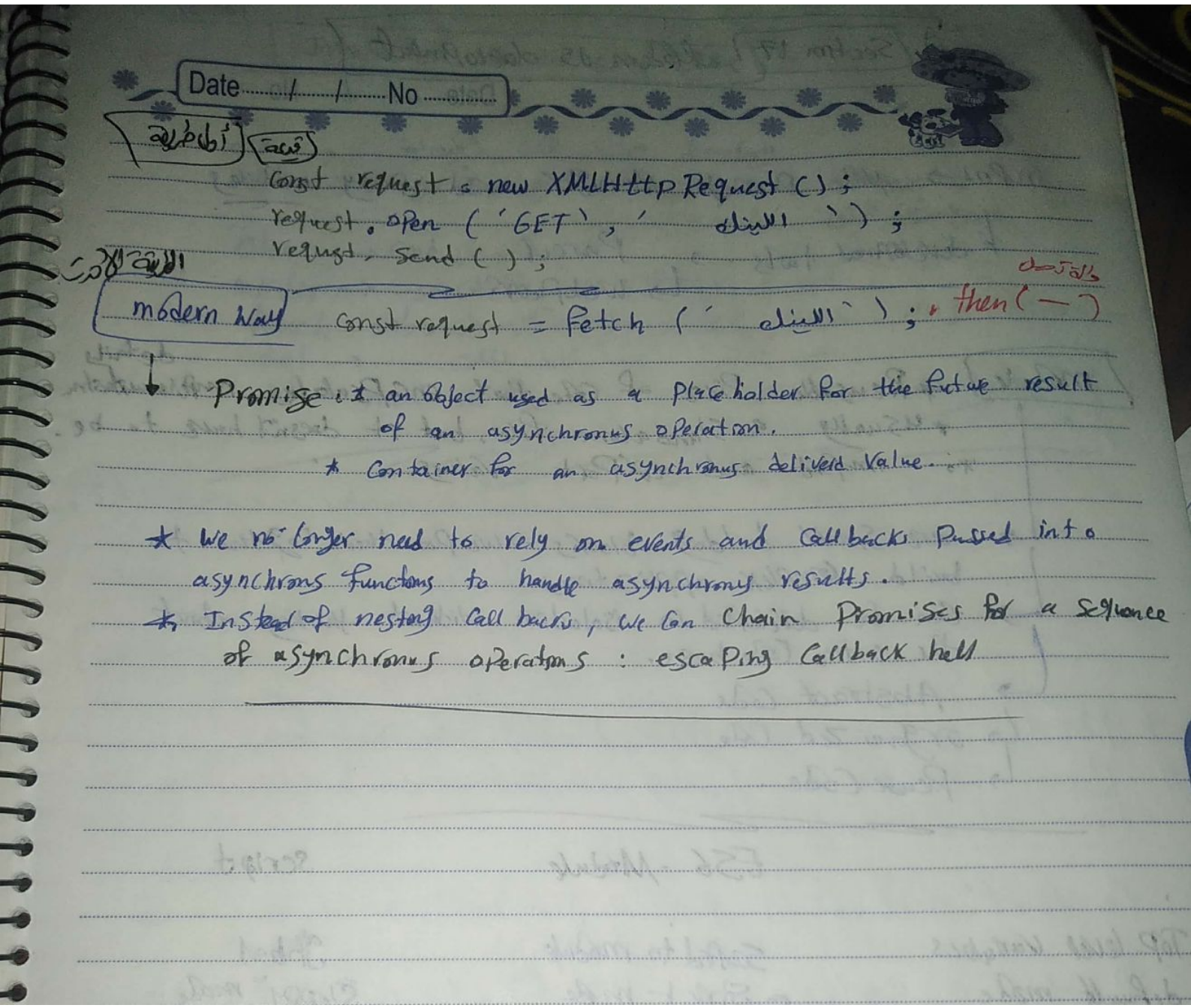
↓

application running on a server, that recieves requests for data, sends data back as response

* We can build our own Web APIs With node-js → there is API for everything Like: Weather data, Data about Countries, Flight data Google Map, Currency Conversion data, Sending mails

* Most Popular data format → JSON data format.

↳ it came with string then we transform it → JSON.Parse (this.responseText)



Section 17 Modern JS development

Date ... / ... / ... No ...

npm → Node Package Manager → 3d party modules

↳ development tools → Parcel
↳ WebPack

module

* Reusable Piece of code that encapsulates implementation details
* Usually a standalone file, but it doesn't have to be.
* import and export values

↳ are small building blocks → We put them together to build complex applications
↳ can be developed in isolation without thinking about the entire code base.
↳ Abstract code
↳ Organized code
↳ Reuse code.

	ES6 Module	Script
Top level variables	Scoped to module	Global
default mode	Strict mode	"Sloppy" mode
Top level this	undefined	Window
Import & export	✓	X No
HTML linking	<script type="module">	<script>
file downloading	Asynchronous	Synchronous

(276)

Introduction to the cmd line

Date ..../..../.... No ....

tab → هيكمل الاسم

dir → محتوى الملف "في اية" directory

cd → change directory → cd .. → يرجع للخلف

cd../.. يرجع خطوتين للخلف

Clear → cmd تمسح الكلام في

mkdir → يعمل ملف جديد → mkdir folder1 في الفولدر

touch → اضافة الكترستيلن مرة واحدة → Ex:- touch M1.JS M2.txt -

del → كنس الملف → del M1.JS

mv → move the file mv M1.JS ... يرجع خطوة للخلف