HTML (Hypertext Markup Language)

ال HTML المسؤولة عن الهيكل الأساسي للموقع

```
<!DOCTYPE html>  ← السطر دا بخبر الـ browser انك بتستخدم Html5
<html>
    <head>
        <title> Try hack me </title>
    </head>
    <body>
        <!-- المحتوى اللي بيظهر في صفحة الويب بيكتب هنا -->  ← comment
    </body>
</html>
```

* `<html>` دا بيعتبر الـ root وده اللي فيه كل حاجة
`<head>` دا بيكتب فيه المحتوى اللي مش بيظهر لليوزر والمحتوى دا ليه اسم اسمه metadata معلومات زي اي
* `<title>` ودا عبارة عن اسم الصفحة اللي بيبقا مكتوب على الـ tab بتاعتك فوق
`<script>` ودا لربط ملفات JavaScript
`<link>` لربط ملفات CSS
`<meta>` تستخدم لتحديد معلومات وصفية تساعد محرك البحث أو تستخدم لتحديد الترميز (encoding)

```
<meta charset="UTF-8">
<meta name="keywords" content="الجيزة, مصر, سياحة">
```

لو كتبت ديت عشان اعرفها اول ما تدخل عبارة عن Keywords فلو حد بحث بكلمة واحدة منهم صفحتي تظهر في محرك البحث

Scanned with CamScanner
تم تصوير بـ CamScanner

اللغة كلها مبنية على الـ tags اللي هو بين < >

attributes عبارة عن معلومات إضافية تضاف إلى العناصر لتوفير تفاصيل أو التحكم في سلوك العنصر أو مظهره
بتم كتابتها بعد الـ tag داخل الـ < >

<tagname attribute="value"> Content </tagname>

العناصر في الـ HTML تنقسم لنوعين

Block-level elements
دي العناصر اللي بتبدأ بـ new line و بتشغل كامل العرض بتاع الصفحة
<h6> ... <h1> ،
 ، ، <p> ،
<form> ،

inline level elements
هي العناصر التي تأخذ المساحة فقط التي تحتاجها بناءً على محتواها
* هذه العناصر لا تبدأ من سطر جديد بل يتم عرضها داخل السطر نفسه
 ، ، <a>
<img> ، ،

body

الـ page بيحتوي على الـ body بيتم وضع على شيء

heading ⇐ من أول <h1> لـ <h6>
الأكبر حجم — الأصغر حجم

<p> ⇐ Paragraph

* الـ attribute اسمها style ممكن استخدمها مع الـ tag ← body
أو h1 أو p بتعمل اي بقا ممكن اغيرلون الخط أو الخلفية

<body style="background-color: red;">
<h4 style="color: blue;"> abdo </h4>

الأقصى

relation

<link rel="stylesheet" href="styles.css">

rel: علاقة او فهم الـ browser ايه نوع الملف اللي مربوطه هنا
stylesheet: دا يعني مربوط ملف CSS
href: يشير إلى مكان الملف

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

script: يستخدم لربط أو كتابة أكواد JS
src: source

base اسمها metadata بردك خاصة بتبع الـ a *

<base href="http://example.com/">

يستخدم لتحديد عنوان URL لجميع الروابط التي لا تحتوي على عنوان URL كامل (أي الروابط النسبية)

 الصفحة

دي مش واخده عنوان URL كامل بتعادي نسبي بعتا وتكمل على الرابط اللي في الـ base

= http://example.com/Page2.html

الأقصى

Scanned with CamScanner
تم المسح ضوئيا بـ CamScanner

بعض الـ tags اللي بيستخدموا على النصوص

 بيكبر الكلمة شويه عن العادي

 بيخلي الخط مائل

 text

دا عباره عن التاب اللي بيظهرلك أول ما تضغط عليه بودلك على اللينك

 text

الـ attribute دي بتخلي اللينك يفتح في tab جديد

عشان اللي انت فيها ما تغيرهاش اللينك يتفتح في نفس الصفحة

بتكتب هنا نص يعبر عن الصورة

عشان لو كان في مشكلة في الصورة او مش عارف يحملها يعرض النص دا.

Lists

UL => unorderd list

```
<ul>
  <li> abdo </li>
  <li> Khaled </li>
</ul>
```

- abdo
- Khaled

list item

OL => ordered list

```
<ol>
  <li> abdo </li>
  <li> Khaled </li>
</ol>
```

1- abdo
2- Khaled

ممكن نغير الترتيب بالأرقام لتحت أو نرتب بالعكس

DL => description list
Dt => term
Dd => description of the term

```
<dl>
  <dt> 1 كلمة </dt>
  <dd> وصف للكلمة 1 </dd>
  <dt> 2 كلمة </dt>
  <dd> وصف للكلمة 2 </dd>
</dl>
```


 => new line

<hr> => بتعمل فاصل افقي في الصفحة

 نص بيستخدموا عشان اركز على نص معين جوه paragraph ولا حاجة

Scanned with CamScanner
تم المسح ضوئيا بـ CamScanner

tables

```
<table>                          → table
  <thead>                        → table row
    <tr>
      <th> Name </th>            → table header
      <th> Age </th>
    </tr>
  </thead>
  <tbody>
    <tr>                         → table data
      <td> abdo </td>
      <td> 15 </td>
    </tr>
    <tr>
      <td> Ibrahim </td>
      <td> 10 </td>
    </tr>
  </tbody>
</table>
```

Name	Age
abdo	15
Ibrahim	10

forms

تستخدم لإنشاء واجهات (أدخال) البيانات في صفحات الويب، وتسمح للمستخدم بإدخال البيانات ثم إرسالها إلى الخادم لمعالجتها

الأقصى

الطريقة اللي هتبعت بيها البيانات — هحط فيه ملف الـ backend اللي هيستقبل البيانات دي

```
<form method="post" action="file.php">
  <div>
    <label for="first">firstname</label>
    <input type="text" id="first" name="first_name"
      placeholder="enter your name">
  </div>
```

placeholder: شايلة نفس جوه خانة بتكتب كدا في المربع بتاع الشكل

name: دا الـ Key أو Variable اللي في قاعدة البيانات وهيتخزن فيها القيمة اللي هتدخلها

* يبقا الـ name دي بيتحط فيها الـ key اللي هيتخزن فيها البيانات اللي هتدخل في قاعدة البيانات

* type ← عشان احدد هو هيدخل ايه text أو email أو password أو message أو number أو date أو submit في آخر

* ايه بقا الـ for والـ id دول: ~~انا بدي للكلمة اللي جنبها~~ جنب المربع اللي قاعد فيه ~~واكتب في الـ input~~ ~~first~~ كدا ~~عامل~~ حاجة في اسم مثلا في الـ id بحيث يكون نفس اللي في الـ for اللي هتدخله

ايه بقا الـ id والـ for دول: الـ id دا حاجة بتميز اللي جواه زي الرقم القومي بتاعك كدا، فأنا بقول للقيمة اللي هتدخل في الـ id بـ first. بيت بقا أربط الـ label بالحاجة اللي هتدخل بيت بـ باستخدم الـ for وحاطيها الـ id بقا وكدا لو أنا جيت ضغط على كلمة firstname هتلاقي المؤشر جوه المربع.

* من غير الـ for بيت أول الـ id لو جيت تضغط على الـ label مش هيعمل حاجة

firstname [enter your name]

الأقصى

```
<div>
    <label> email </label>
    <input type="email" name="email">
</div>
<div>
    <label> message</label>
    <textarea name="message"></textarea>
</div>
```

email

message

بيديك مربع كبير ممكن تكبر حجمه عادي بالماوس
مثلا انا بكتب رسالة مثلا
واذا كلام كتير

```
<div>
    <label> gender </label>
    <select name="gender">
        <option value="MALE"> male </option>
        <option value="FEMALE"> female </option>
    </select>
</div>
```

select يتيح للمستخدم اختيار خيار واحد من عدة خيارات
Value تحدد القيمة المرسلة اذا تم اختيار الخيار دا
gender في المتغير MALE هتبقا Value فال male يعني لو اخترنا
هتخزن فيه MALE

```
<div>
    <label> age </label>
    <input type="number" name="AGE" value="25">
</div>
```

لو معتبرتبطش فيها default الصفحة
أول مايوزر يفتح الصفحة هيلاقي 25 مكتوبة في الخانة
بقا يمسحها ويكتب حاجة تانية

```
<div>
    <label> birthday </label>
    <input type="date" name="BIRTH">
</div>
<input type="submit" value="submit" name="SUBMIT">
```

لما يعدد الأسم اللي هيظهر
على الزر

```
</form>
<a href="لينك">
<img src="photo.jpg" alt="__">
</a>
```

كدا انا عملت صوره لو ضغط عليها هتوديني للينك اللي كتبه

```
<blockquote cite="اكتب لينك مصدر الاقتباس">
```

الاقتباس

النص اللي بيكتب لنا بيبقا
احسن ومنظم عكس

```
</blockquote>
```

اللينك اللي cite دا مش بيظهر على الصفحة عادي
ممكن تبقى مهمة لمبرمجي web developer
لانه بيفيد محركات البحث (SEO) في تعزيز مصداقية موقعك
وعادي بقا نظهر اللينك دا في الصفحة بردك باستخدام CSS أو JS

```
<p>
    The <abbr title="world wide web"> www </abbr>
    is awesome
</p>
```

The www is awesome

لو وقفت على www هيظهرلك تعرفها world wide web

Scanned with CamScanner
تم المسح ضوئيا بـ CamScanner

Variables

يوجد ثلاث طرق لتعريف المتغيرات Const ، Let ، Var

Scopes :-

1. Global Scope عبارة عن المتغيرات التي يتم تعريفها خارج أي fun ، ويمكن الوصول إليها من أي مكان
2. Block Scope :- المتغيرات المعرفة باستخدام let و const تكون
   - داخل نطاق الـ Blocks المعرفة فيه بس زي if أو for
   - الـ Var مش زيهم
3. Function scope :- المتغيرات المعرفة باستخدام Var داخل الـ fun
   تكون داخل نطاق الـ fun دي بس

* كدا الـ let والـ const نطاقهم هم اللي مكتوبين فيه يعني لو محطوطين في fun بس برة if أو for مثلا يبقا كدا هم نطاقهم الـ fun
لو محطوطين في if أو for يبقا دول نطاقهم لو برة fun أو أي Block زي if
هم كدا يعتبروا Global على اللي مكتوبين فيهم بس

* الـ Var بقا لو برة أي حاجة تبقا Global على أي ملف
لو جوة fun تبقا نطاقه هي الـ fun دي
لو جوة Block زي if مثلا نطاقها مش هيبقا if زي let كدا و هيبقا نطاقها
الحاجة اللي فيها الـ if دي يعني لو fun تبقا نطاقها الـ fun دي لو في الـ Global
تبقا نطاقها الـ Global

الأقصى

Var

الـ Variables المعرفة بـ Var يمكن إعادة تعيين قيمة المتغير ، ويمكن
وإعادة تعريفها باستخدام Var في نفس الـ scope تاني

```
Var name = "abdo";
name = "Ahmed";   ✓ reassigned
```

```
Var x = 25;
Var x = 20;   ✓ re-declares
```
والـ x بقت بـ 20

Let

عادي تعمله reassigned بس مينفعش تعمله re-declares في نفس الـ
error يديك Scop

Const

مينفعش تعمله reassigned عشان هو constant ومينفعش تعمله
scope في نفس الـ re-dec
* لازم وانا بعرف المتغير الـ const أديله قيمة لو مدتهوش يديك error

الأقصى

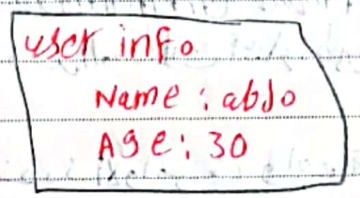
Scanned with CamScanner
تم تصوير بـ CamScanner

7 console.groupEnd() , console.group()

يستخدم لتجميع الرسائل في مجموعات ، يمكن للمطورين استخدامها لتنظيم الرسائل بشكل هيكلي داخل الـ console

```
console.group("user info");
console.log("Name : abdo");
console.log("Age : 30");
console.groupEnd();
```

عنوان المجموعة group



8 console.timeEnd() console.time()

يكتب بينهم كود عاوز اعرف بيستغرق قد أي للتنفيذ

```
console.time("time taken");
// some code to measure
console.timeEnd("time taken");
```

الاسم اللي عاوز اعرف بيستغرق قد ايه

عشان اميزه

عشان لما اجي انهيه

ينتهي

استخدم نفس الاسم فيعرف انك عاوز الوقت بين الأتنين اللي فيهم time taken

* يبقا باستخدام الـ console يمكنك تتبع سير الكود خطوة بخطوة ومعرفة أين يمكن أن يحدث الخطأ أو التأخير

* الـ console أداة أساسية أثناء الـ debugging في الـ JS

Scanned with CamScanner
تم تصوير بـ CamScanner

JAVASCRIPT

High level, interpreted programming Language

Console : هو واجهة تتيح للمطورين التفاعل مع Programming environment من خلال الكتابة والقراءة في الـ Command Line
يستخدم بشكل رئيسي في الـ browsers والـ Development environments لتشغيل الأوامر، طباعة الرسائل، مراقبة الأخطاء، وحل المشكلات أثناء تطوير التطبيقات

* الـ console هو عبارة عن object يتيح لك تنفيذ الأوامر في واجهة المستخدم (interface) وتحديداً في console الخاص بالمتصفح أو بيئة التشغيل (مثل Node.js) ويتم استخدامه بشكل واسع لتحليل الكود وتتبعه أثناء عملية التطوير

وظائف الـ console :

1. console.log( )
يستخدم لطباعة رسائل إلى الـ console. هذه الرسائل يمكن أن تكون string أو numbers أو objects

```
console.log("Hello, world");
```

2. console.error( )
يستخدم لطباعة رسائل الأخطاء، يكون النص المطبوع غالباً باللون الأحمر لتمييزه

```
console.error("this is an error message");
```

3. console.warn( )
يستخدم لطباعة الـ warning باللون الأصفر

4. console.info( )
لطباعة معلومات غير مهمة ولكن مفيدة للمطورين، باللون الأزرق

5. console.debug( )
يستخدم لطباعة رسائل لتتبع الأخطاء أو فحص الكود

6. console.table( )
يستخدم لطباعة objects أو arrays بشكل منظم على شكل جدول

Scanned with CamScanner
تم المسح ضوئيا بـ CamScanner

Scanned with CamScanner

عشان اجزء محتوى حاجة كدا حاجة برضه <section> مثلا معايا
مثال ممكن اعمل عنوانها لوحده في section واعمل محتواها في section
تاني والخاتمة في section
الخلاصة بستخدمه لو عايز اقسم حاجة ليها علاقة ببعض زي هنا كدا
عنوان ومحتوى وخاتمة كلهم لنفس المقالة

```
<section>
    <h2> ——— </h2>
    <p> ——— </p>
</section>
<section>
    <p> ——— </p>
    <p> ——— </p>
</section>
```

عشان الكود يبقا منظم بنقسم الـ body لـ 3

header
بيحتوي على المعلومات العلوية بتاعت الصفحة زي الـ logo مثلا

main
بيحتوي على المحتوى بتاع الصفحة

footer
بيحتوي على المعلومات اللي بتكون في نهاية الصفحة زي الـ copy right

* في حاجة اسمها id بستخدمها عشان اسمي عنصر معين مثلا

```
<p id="par1"> ——— </p>
```

كدا لو حبيت اعمل اي حاجة للباراجراف دا
استخدم الـ id بتاعه واحط قبل #

لو انا حط كدا لينك في الصفحة جوه بعض بقوله حاجة اسمها <nav> عبارة عن حاجة مثلا بجمع فيها اللينكات — navigation links

```
<nav>
    <a href="——"> ——— </a>
</nav>
```

* ممكن مثلا لو الصفحة فيها مقالات أو حاجة أقسم كل مقالة بحاجة اسمها <article>

```
<article>
    <h1> ——— </h1>
    <p> ——— </p>
</article>
```

Arrays

* ممكن تحتوى على أنواع بيانات مختلفة
* Dynamic أى يمكن تعديل حجمها أثناء التنفيذ (إضافة عناصر أو حذف)

```
let fruits = ["apple", "banana", "cherry"];
Console.log(fruits[1]); // banana
fruits[0] = "abdo"
console.log(fruits); // ["abdo", "banana", "cherry"]
Fruits[3] = "apple"        كده هتضيف apple للآخر
fruits.push("orange");
console.log(fruits); // ["abdo", "banana", "cherry", "apple", "orange"]
```

1. `push()` تضيف عنصر للنهاية أو أكتر وبترجع بال length الجديد
2. `unshift()` تضيف عنصر للبداية أو ممكن تضيف أكتر من عنصر وبترجع بالـ length الجديد
3. `pop()` لإزالة العنصر الأخير — كمان عادى ترجعه وتحطه فى قيمه
4. `shift()` لإزالة العنصر الأول

```
let last = fruits.pop();
    ↳ orange
fruits.pop();
console.log(fruits); // ["abdo", "banana", "cherry"]
```

وكدا آخر عنصر اتمسح

5. `splice(start, delete count, الباقى لو عاوز تضيف)`

```
let fruits = ["apple", "banna", "cherry"];
fruits.splice(1,1);
Console.log(fruits); // ["apple", "cherry"]
fruits.splice(1,0, "bannana");
console.log(fruits); // ["apple", "banana", "cherry"]
```

- هيبدأ من العنصر 1 وهيمسح الكلمة وعالما هتضيف بيقا الـ delete count بصفر
- هيبدأ من الـ index 1 ويشيل خانه واحده
- لو كان مثلاً بواحد كان هيمسح عنصر وهيحط الجديد

6. `slice(start, end)` عشان آخد sub array من الـ array

7. `concat()` تجمع بين كذا array وترجع بـ array واحده

```
let x = ["a", "b"];
let y = ["d", "f"];
let combines = x.concat(y);
    ↳ ["a", "b", "d", "f"]
```

8. `indexOf(item)` بترجع بالـ index بتاع الـ item لو لقته
ولو لا بترجع بـ -1

9. `include(item)` true ← لو موجود
false ← لا

10. `reverse()` تعكس الـ Array

11. `join(sperator)` بتدمج الـ array لـ string

```
let x = ["ap", "cd", "ef"];
let joines = x.join(", ");
Console.log(joines); // "ap, cd, ef"
```

12. `sort()` بترتب الـ array

* لو الـ Array عندى const عادى اضيف فيها واحذف بس مينفعش ادوح احط فيها كلها قيم جديدة

```
const arr = [1, 2, 3]
arr = [1, 2, 3]   X   مينفعش
```

عادى لما اعمل اى حاجه بس مينفعش احط فيها array جديدة

Scanned with CamScanner
تم المسح ضوئيا بـ CamScanner

6. string.includes("—") تفحص الي قبلها بس لنا بترجع true لو لقت الكلمة و false لو لا

7. string.trim() بتزيل الفراغات من بداية ونهاية النص

8. string.replace(searchValue, newValue) بتستبدل أول تطابق

9. string.replaceAll(searchValue, newValue) بتستبدل كل التطابقات

10. string.split("—") تقسم الـ string إلى Array بناءً على محدد معين

```
let text = "a,b,c";
let x = text.split(",");   // بتقسمه بناءً على ,
console.log(x);   // ["a","b","c"]

console.log(text.split("")); // بتقسم الـ string لحروف
// ["a",",","b",",","c"]
```

```
let text = "abdo";
let reverses = text.split("").reverse().join("");
console.log(reverses);   // odba
```

أول حاجة عملتها حولت الـ string إلى array
وبعدين عكست الـ array وبعدين جمعتها تاني

* مفيش reverse في الـ string علشان كدا حولت الـ string الأول array

---

```
let sentence = "javascript is a greet language!";
let words = sentence.split(" ");
console.log(words.length);   // 5
```

الأقصى

Scanned with CamScanner
الماسح ضوئي CamScanner

```
let x = null;          typeof object
x = 60;
let y;
let z = undefined
Console.log(y);    // undefined
Console.log(z);    // undefined
Console.log(x)     // 60
Console.log(typeof y);   undefined
            z);
let w = null;
Console.log(w);        null
Console.log(typeof w);   object
```

String

Concatenation

```
let name = "abdo";
let age = 25;
let message = "my name is " + name + " and my age is " + age;
```

template string

```
let message = `my name is ${name} and my age is ${age}`;
```

Backticks بتستخدم مش بتستخدم template ال G*

الأقصى

String زي باقي الـ arrays في الحروف immutable ←

```
let text = "Hello";
Console.log(text[0]);   // H
text[0] = z;
Console.log(text)   // Hello
```

مش هيعدل حاجة وهيسيب زي ما هي

```
let x = "";
Console.log(text.length);   // 5
```

length ← property

لو كتبت فعل string فاضية

Built in Fun

1. string.toUpperCase() تحول الـ string ← Capital
2. string.toLowerCase() ← small
3. String.slice(start, end) بيقتطع جزء من الـ string
   - end ← default النهاية

```
let text = "Hello, world";
Console.log(text.slice(0,5));   // Hello
Console.log(text.slice(7));     // world
```

* لو كتبت بالسالب index جوه مثلاً -2 ← index ابدأ من آخر L بقا

```
Console.log(text.slice(-3));   // rld
```

ابدأ من عند 3- من الآخر

4. string.substring(start, end) زي slice بس مش بتاخد قيم سالبة
5. string.indexOf("كلمة بتبحث فيها") بترجع بالـ index بتاع أول حرف في الكلمة
   - ولو مش موجودة ترجع -1

الأقصى

Scanned with CamScanner

Data types

~~String, Numbers (integer, float), Booleans~~

Number @ 5, -10, 3.5 ← Primitive data types [1]

```
let x = 5;
let y = 3.5;
```

String [2] بتتعرف بـ "" أو ''

False or true Boolean [3]

undefined [4]

cat

- Variable that has been declared but not assigned a value is undefined

عبارة عن Variable عرفته وادتيله undefined أو مدتهوش حاجة فالت

```
let user;
Consol.Log(user);   // undefiend
let x = undefiend;
```

لأما مدتهوش قيمة فالت هتروح الـ default بتاعها undefiend

أو انا ديله قيمة undefiend

Bigint [5] يمثل الأرقام الكبيرة التي لا يمكن تمثيلها باستخدام Number

null [6] يمثل قيمة فارغة

لكن يعتبر نوعه object بسبب خطأ في اللغة زمان

undefined , null الفرق بينهم

~~من حيث النوع~~ ال null ← object

undefined ← undefined

* ال null يتم تعيينها يدويًا

ال undefined يتم تعيينها ~~تلقائيًا~~ افتراضيًا عندما لا يتم إعطاء قيمة للمتغير

* null قيمة فارغة

undefined قيمة غير محددة

* عادي الاتنين بعد كدا اعمل فيهم قيم

Symbol [7] عندما يكون لديك object يحتوي على العديد من الخصائص وقد تحدث مشاكل إذا كان هناك key بنفس الاسم في مكانين مختلفين

فال symbol بيخلي الـ key يتم إنشاؤه يكون unique

Non primitive (Reference) data type [2]

object [1]

Represents a collection of key-values pai

```
Let person = {
    name : "abdo",
    age : 20 ,
    city : "Egypt"
};
```

Arrayes [2]

```
Let numbers = [1,2,3,4];
```

function [3]

Date [4]

↓

Built-in-object

```
Let today = new Date();
```

↳ object وكدا

Date من النوع

```
Let name = "abdo" + " Khaled";      string
Let age = 25;                       number
Let Length = 175.5;                 number
Const country = "Egypt";            string
// Const a;   error                 لازم تديله قيمة
Var b = 5;                          number
Var b = 10;   ✓
Let money;   → typeof → undefined
money = 55;  → number
```

Scanned with CamScanner

* لو استخدمت for in مع Array هيعتبر الـ index دا Variable الـ array مش العناصر ذات نفسه اللي في الـ array كأنه مستخدم for عادية

```
Cons arr = [10, 55, 60];
for ( let index in arr ){
    console.log( index, arr[index])
}
```

// 0 10
// 1 55
// 2 60

for - of loop

is used to iterate over iterable objects like arrays, Strings, maps or sets.

```
Const arr= [1, 2, 6, 10];
for (let itr of numbers){
    console.log(itr);
}
```

// 1
// 2
// 6
// 10

method ← for each

```
array.forEach( function (element, index, array){
});
```

مش لازم نفس اسم الـ array بس لازم كدا كدا هيتخزن فيها الـ array

element * العالي item هو الـ array اللي شغال عليه

index * هو رقم الـ index للعنصر (optional)

array * الـ Array نفسها (optional)

```
const numbers = [5, 6, 7];
numbers.forEach (function (num){
    console.log (num);
});
```

// 5
// 6
// 7

```
let numbers = [1, 2, 3];
numbers.forEach (function(number, index, arr){
    arr[index] = number * 2;
});
console.log (numbers); // [2, 4, 6]
```

b-

```
let obj = {
    id: 50,
    firstname: "abdo"
};
object.keys ( obj ).forEach (function (key){
    console.log (key, obj[key]);
});
```

// id 50
firstname abdo

* the forEach method does not support the break statement or return

في حاجة بتلوب على كل العناصر لو انت مش عاوز كدا وبترجع بـ undefined

```
const numbers = [1, 2, 3, 4, 5];
const result = numbers.forEach (function (num){
    if (num == 3){
        return;
    }
    console.log (num);
});
console.log ( result ); // undefined
```

return → مش بتوقف الـ forEach لكن بتعمل skip الخطوة دي بس

// 1
// 2
// 4
// 5

Scanned with CamScanner
الملف ممسوح ضوئيا بـ CamScanner

Loops

1] for Loop

```
for(let i=0 ; i<5 ; i++){
    console.log(i);
}
```

زي C++ عادي

2] while Loop

```
let j=0
while (i<3){
    console.log
    j++;
}
```

زي C++

3] do while Loop

```
do{
    // code
}while(condition);
```

زي C++

4] for in Loop

is used to iterate over the properties of an object

```
for (variable in object){
    // code
}
```

variable هيبقا هوا الـ key بتاع الـ loop

```
const person={
    firstname : "Ahmed";
    age : 25
};
for(let key in person){
    console.log(key , person[key]);
}
// firstname Ahmed
// age 25
```

الأعضاء

Scanned with CamScanner
تم المسح ضوئيا بـ CamScanner

~~Strict~~

Shorthand for methods

You can define methods in objects without using the function keyword.

```
let animal = {
    type: "Dog",
    Sound() {
      console.log("woof!");
    }
};
animal.Sound(); //woof!
```

* لو الـ method واخدة parameter بعرفها جوا ()

```
const obj = {
    myname: "abdo",
    age: 25,
    address: {
      country: "EGYPT",
      city: "mansoura"
    },
    arr: [1,2,3,"hhhh"],
    greet: function() {
      console.log("Hello, " + this.myname + "!");
    },
};
console.log(obj.arr[3], obj["age"]); // hhhh 25
console.log(obj.address.city)        // mansoura
obj.greet();  // Hello, abdo!
```

الأفضل

نفس نوع الـ object

```
const {
    myname,
    arr,
    address: {city},
} = obj
```

destruct

بساويه بالـ object

كل key مكتوبة زي اللي في الـ object

تبقى variable متخزن فيها الـ value اللي في الـ object دا

```
console.log(myname, city, arr[3]) // abdo mansoura hhhh
```

delete obj.age كدا حذفت الـ key اللي اسمها age

```
const ids = [
    {
      id: 1,
      who: "abdo"
    },
    {
      id: 2,
      who: "ahmed",
    }
];
console.log(ids[0].who); // abdo

const tojson = JSON.stringify(ids);
console.log(tojson);
// [{"id":1,"who":"abdo"},{"id":2,"who":"ahm
```

* البيانات بتتنقل غالبًا باستخدام json دا للـ server

وغالبًا شكله عامل زي كتابة الـ object بالظبط باختلاق بسيط الـ key بين " "

الأفضل

Scanned with CamScanner
تم المسح ضوئيا بـ CamScanner

object literal

syntax:

```
Let objectName = {
    key1 : value1,
    key2 : value2,
    key3 : value3
}
```

دي كدا على بعض → property: حاجة اسمها key والـ value اللي جاية بعد الـ key

عشان تعمل ACCESS عليه

```
objectName.key
or
objectName["key"]
```

* ممكن تعمل define لـ fun كمان جوا الـ object اللي في الحالة دي بيقا اسمها مش fun اسمها method

Computed Property Names:

You can use dynamic property names inside an object literal using square brackets.

```
Let prop = "dynamickey";
Let obj = {
    [prop] : "this is a dynamic value"
};
console.log(obj.dynamickey); // this is a dynamic value
```

Shorthand for property

if the property name matches the variable name, you can omit the key and just use the variable.

```
Let firstname = "Alice";
Let lastname = "smith";
Let user = {firstname, lastname};
console.log(user); // {firstName: 'Alice', lastName: 'smith'}
```

الأفضل

Scanned with CamScanner
تم المسح ضوئيا بـ CamScanner

Constructor function

is a special type of function used to create and initialize objects. it acts as a blueprint for creating multiple objects with the same structure and behavior
the process of creating an object using a constructor fun is called instantiation

* يفضل ان الـ constructor يبدأ بـ capital

* It is called using the `new` keyword.

```
function constructor_Name (parameters) {
    // intialize properties
    this.property1 = value1;
    this.property2 = value2;
    // Add methods (optional)
}
```

* this تستخدم لتعريف الـ properties بتاعت الـ object
* new تستخدم لكي تنشئ object جديد وتعيد الـ properties والقيم الجديدة وترجع بالـ object الجديد ده

```
function Person (Firstname, Lastname, dob){
    this.firstName = Firstname;
    this.LastName = Lastname;
    this.Dob = new Date (dob)

    this.getFullname = function() {
        return `${this.firstName} ${this.LastName}`;
    }
}

// instentiate object
const person1 = new Person('abdo', 'khaled', '3-31-2004');
const person2 = new Person('mary', 'smith', '3-6-2005');
```

```
console.log(person1.fullName());  // abdo khaled
console.log(person1.Dob);         // wed Mar 31 2004
console.log(person2.Dob.getFullYear());  // 2004
console.log(person2);             // يطبع الـ object كله تقريبا
```

* الـ Date: هو عبارة عن constructor function ويستخدم لإنشاء objects تمثل تاريخاً ووقتاً معيناً
وجواه methods بتاعته زي الـ getFullYear() getFullName getSeconds() getMinutes() getHours() getMonth() setDay()

* لما باجي اعمل object من constructor function فتلقائي بيتعمل للـ object ده object تاني جواه اسمه prototype والـ object ده بيحط فيه properties أو methods تحت الـ object المبدئي اللي معمول من الـ constructor

* بستخدم الـ prototype عشان اوفر memory شويه أزاي؟
بدل مثلاً في المثال اللي فات دا عملت method اسمها getFullName والـ method دي بتتخزن في كل object بيتعمل جديد يعني هي اتعملت في person1 و person2 & دا انا اروح اعملها في الـ prototype وكدا تعملها مرة واحدة وكل الـ object تبقى شايفاها بردك.

* كل الـ methods والـ properties اللي تضيفها لـ الـ prototype هتكون مشتركة بين جميع الـ objects اللي تم إنشاؤها من نفس الـ constructor
ودا معناه إن كل الـ objects مش هتحتاج تعمل الـ methods دي مرة تانية
والي تخزين نسخة منها في كل object

* يتم تخزين نسخة واحدة فقط في الـ prototype وبالتالي يتمكن كل الـ objects من الوصول لنفس الـ method دون تكرارها

* كل constructor ليه prototype ودا فقط يعني مش كل ما اعمل object بيتعمل prototype ليه constructor & دا كل constructor ليه prototype واحد فقط ~~لكل~~ كل الـ objects وبينها مشترك بين كل الـ objects اللي معمولة من الـ constructor دا

Condition

```
if (condition1){
}
else if (condition2){
}
else {
}
```

```
switch (value){
  case case1:
    break;
  case case2:
    break;
  default:
}
```

Ternary operator

```
condition ? if true : if false ;
```

function

1 الطريقة العادية

```
function function_Name (parameters){
}
```

2 Function Expression

يتم تخزينه الـ fun في Variable

```
const function_Name = function (parameters){
}
```

مثال

```
const greet = function (name){
  console.log("Hi," + name + "!");
};
greet("abdo");          // Hi, abdo !
```

3 Arrow function

عبارة عن صيغة مختصرة لكتابة الـ fun باستخدام =>

```
const function_Name = (parameters) => {
};
```

* لو الـ fun بتاع الـ return حاجة واحدة زي كدا مش لازم اعمل { } ومش لازم أكتب return

```
const mul = (a,b) => a*b;
console.log(mul(4,2));   // 8
```

```
= (a,b) => { return a*b; };
```

* لو مفيش غير parameter واحد مش لازم اعمل ( ) أكتبه طالول

filter ( )

عبارة عن method تستخدم لاستخراج عناصر من Array في حالة تحقق شرط معين، تقوم بإنشاء array جديدة تحتوي على العناصر التي مرت عبر الفحص، ولا تؤثر على الـ array الأصلية

```
Const newArray = array.filter(function(currentValue, index, array){
});
```

optional

```
Let numbers = [1,2,3,4,5];
Let even = numbers.filter(function(num){
    return num%2===0;
});
console.log(even);   // [2,4]
```

```
Let even = numbers.filter(num => num%2===0);
```

ممكن اعمل كدا

Parameter الـ Arrow Function دي حاجة اسمها
ثم القيمة اللي الدالة هترجع بيها ويعديه يعمل =>
لو أكتر من حاجة أو هطبع وكدا لازم اعملها {}

== يقارن بين القيم بعد تحويل الأنواع تلقائيًا إذا كانت الأنواع غير متطابقة
=== يقارن القيم مع الحفاظ على النوع (Type-safe comparison)

```
5=='5'   true
5==='5'  false
```

صاحب الـ '5' هيتحول لرقم ويعديه قارنه بالرقم

```
const users = [
  { name: "John", age: 25 },
  { name: "Jane", age: 30 },
  { name: "Jim", age: 35 }
];
const adults = users.filter(user => user.age >= 30);
Console.log(adults);
```

[{name:"Jane", age:30}, {name:"Jim", age:...}]

```
const adults = users.filter(function(user){
    return user.age >= 30;
}).map(function(user){
    return user.name;
});
Console.log(adults);  // ["Jane", "Jim"]
```

الأقصى

Scanned with CamScanner
تم المسح ضوئيا بـ CamScanner

map()

the map() method is used to create new array by applying a given function to each element of an existing array. unlike forEach, which doesn't return a new array

* map return a new array containing the result of the callback function for each element

```
const newArray = array.map(function(currentValue, index, array){
                                                    Optional
//
})
```

* one-to-one mapping: each element in the original array is mapped to one element in the new array

---

```
const numbers = [1,2,3];
const doubled = numbers.map(function(num){
    return num*2;
});
console.log(doubled); // [2,4,6]
console.log(numbers); // [1,2,3]
```

```
const users = [
  { name: "John", age: 25}
  { name: "Jane", age: 30}
  { name: "Jim", age: 35}
];
const userAges = users.map(function(user){
    return user.age;
});
console.log(userAges); // [25,30,35]
```

Multiple element

① document.getElementsByClassName( )
* selects all elements with a specific class name.

```
<div class="box"> first </div>
<div class="box"> second </div>

const boxes = document.getElementsByClassName("box");
console.log(boxes.length)  // 2
```

② document.querySelectorAll( )
selects all element that matches a CSS selector
.class / #id / tag

```
const boxes = document.querySelectorAll(".box");
```

③ document.getElementsByTagName( )
selects all elements with a specific tag name (div, p)

```
const boxes = document.getElementsByTagName("div");
```

```
<ul class="items">
  <li class="item"> item 1 </li>
  <li class="item"> item 2 </li>
  <li class="item"> item 3 </li>
</ul>

const ul = document.querySelector(".item");

//ul.remove();
//ul.lastElementChild.remove();
//ul.firstElementChild.textContent = "abdo";
//ul.children[1].innerText = "abdo";
ul.children[0].innerHTML = '<h1> Hello </h1>';

element.style.color = "red";
element.style.backgroundColor = "blue";
```

document DOM (Document object Model)

هو عبارة عن object تابع لـ window ويمثل صفحة الـ HTML للصفحة
* هو النقطة الرئيسية للوصول لمحتويات الـ HTML ومعالجتها والتعديل عليها

properties of document

document.title	تعرض أو تعدل عنوان الـ Page الموجودة في <title>
document.body	محتوى الـ body
document.URL	تعرض رابط الصفحة الحالي
document.forms	تعرض List بكل الـ forms الموجودة في الـ Page

methods

getElementById(id)	اختيار عنصر باستخدام الـ id (بتاع العنصر)
querySelector(selector)	اختيار أول element يطابق محدد CSS معين
querySelectorAll(selector)	اختيار جميع الـ elements التي تطابق محدد CSS معين
createElement(tag)	
getElementByClassName(class list)	اختيار كل العناصر التي تحمل الـ class list

Single Element

لو عايز الحصول على single element
1. getElementById() باستخدام الـ
تستخدم للوصول لأي عنصر معين باستخدام الـ id بتاعه

```
<h1 id="title">abdo</h1>

let element = document.getElementById("title");
console.log(element);          // <h1 id="title">abdo</h1>
console.log(element.textContent);   // abdo
```

2. querySelector() باستخدام الـ
تستخدم لتحديد أول عنصر يتطابق مع محدد CSS (مثل class أو id)

```
<p class="text">first</p>
<p class="text">second</p>

let element = document.querySelector(".text");
console.log(element.textContent);   // first
```

الـ DOM هو عبارة عن API تستخدم للتعامل مع الـ HTML
DOM Tree بحيث إنه بيتم تحويل صفحة الـ HTML إلى شجرة هرمية تسمى
بحيث كل عنصر في صفحة الـ HTML (مثل <p>) يصبح Node
الـ DOM Tree في
الـ root بتاع الـ DOM Tree هو <html>

```
function Person (firstname, lastname, dob) {
    this.firstName = firstname;
    this.lastName = lastname;
    this.Dob = new Date (dob);
}
Person.Prototype.getfullName = function() {
    return `${this.firstName} ${this.lastName}`;
};
const person1 = ____________
const person2 = ____________
const person3 = ____________
console.log (person2.getfullName());
```

طبيعي عادي جداً في المثال الأول

```
console.log (person2);
```

هتلاقي جواه الـ firstname و الـ lastname و الـ Dob

وممكن هتلاقي الـ getfullName بس هتلاقي فيه بروتوتايب ↑ انتقوا هتلاقي الـ

---

class

```
class className {
    constructor (parameters) {
        this.property1 = value1;
        this.property2 = value2;
    }
    methodName() {
        console.log (_____);
    }
}
```

كملت constructor بيدل الـ constructor function بيدل فيه الـ properties + هنا كدا الـ methods اللي خارج الـ constructor بتيجي الـ prototype

```
class Person {
    constructor (firstname, lastname, dob) {
        this.firstName = firstname;
        this.lastName = lastname;
        this.Dob = new Date (dob);
    }
    getfullName() {
        return `${this.firstName} ${this.lastName.}`
    }
}
const person1 = new Person (_____);
    person2 = _____
console.log (person2.getfullName());
```

Scanned with CamScanner