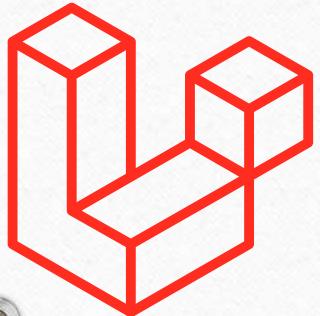


MySQL



Introduction to back end with PHP



Day 2: Introduction to Full stack track with php

Eng. Ahmed Mohamed Abu-Bakr

Full stack web developer Laravel-react





Day 2: websites



Browser



Server



Database



الجزء المسؤول عن جانب العميل يسمى ب client side والكود الذي يعمل فيه يتوقف على ال browser ويقوم بعمل مطوري الواجهة الامامية front-end developers ويتميز بالسرعة وقلة التعقيد وهو مسؤول عن الشكل النهائي للموقع

الجزء المسؤول عن جانب البيانات او التعامل مع السيرفر يسمى ب server side والكود الذي يعمل فيه يتوقف على ال server وطريقة ارسال واستقبال البيانات ومن خلاله يتاثر سرعة الموقع



Day 2: Front-end simple structure

مثال توضيحي على الفرق بين أهمية اللغات الثلاثة
الرئيسية المكونة ل واجهة المواقع

HTML



JS



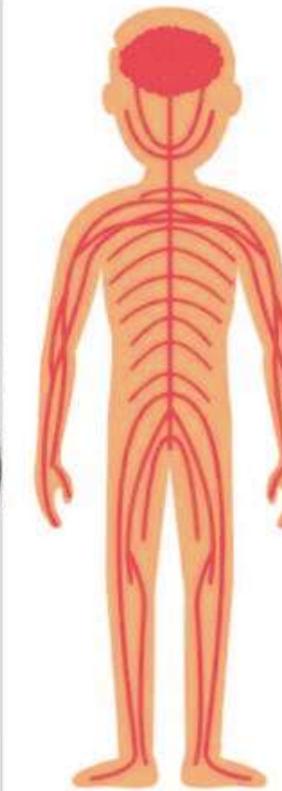
CSS



HTML



JS



CSS



1 Syntax to write an HTML element

Opening Tag

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

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

Closing Tag

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

```
<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

Attribute and its value (optional)

Attributes are like options of an element. Attributes have value.

Children (optional)

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

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> **

<a>

<div>

lists:

** **

Special formatting

<blockquote> <pre>

multimedia:

** <video>**

<audio>

separators:

**<hr />
**



Day 2: HTML (Hyper Text Markup Language)

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>
```

أهم العناصر

Code Formatting

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

دائماً أي صفحة HTML بتبدأ بهيكل ثابت.
العناصر دي أساسية: <!DOCTYPE html>, <html>, <head>, <body>.
داخل <head> بنحط الميتا داتا، العنوان، روابط CSS.
داخل <body> يكون المحتوى الظاهر للزائر.

1. <html> : العنصر الرئيسي الذي يحتوي على كل المحتوى.
2. <head> : يحتوي على معلومات عن الصفحة مثل العنوان، الروابط إلى CSS).
3. <body> : يحتوي على المحتوى المرئي للصفحة.
4. <h1> <h6> : عناوين بمستويات مختلفة (من الأكبر إلى الأصغر) : إلى
5. <p> : فقرة نصية.
6. : رابط إلى صفحة أخرى.
7. : لـ إدراج صورة (وصف المـ صورـة).
8. , , : قوائم غير مرتبة ومرتبة.
9. <table> <td> <tr> : للـ خلايا للـ صفوف و للـ جداول.
10. <form> <input> <button> : للأـ زارـاـن للـ حقول و لـ نماذـجـ الإـ دخـالـ.



Day 2: HTML (Hyper Text Markup Language)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>My First Page</title>
  </head>
  <body>
    <h1>Welcome!</h1>
    <p>This is a paragraph.</p>
  </body>
</html>
```



دائماً أي صفحة HTML تتبدأ بهيكل ثابت.

العناصر دي أساسية: <!DOCTYPE html>, <html>, <head>, <body>. داخل <head> بنحط الميتا داتا، العنوان، روابط CSS. داخل <body> بيكون المحتوى الظاهر للزائر.

العنصر الرئيسي الذي يحتوي على كل المحتوى : 1. <html>.

2. <head> مثل العنوان، الروابط إلى) يحتوي على معلومات عن الصفحة .

3. <body> يحتوي على المحتوى المرئي للصفحة .

4. <h1> <h2> <h3> <h4> <h5> <h6> عناوين بمستويات مختلفة (من الأكبر إلى الأصغر) :

5. <p> فقرة نصية .

6. رابط إلى صفحة أخرى : <a> .

7. لإندراج صورة : .

8. , , قوائم غير مرتبة ومرتبة : ,

9. <table> للصفوف و <tr> مع لإنشاء جداول : <table> . (للخلايا <td>)

10. <form> للأزرار للحقول و <input> مع لنماذج الإدخال : <form> . (<input>)



Day 2: Task 1 (3 minutes max)



Task 1 (3 minutes max)

أنشئ ملف HTML يحتوي على:

- عنوان الصفحة في <title> جواه اسمك و جنبها my first page
- عنوان <h1> داخل <body> اهلا اسمي (اسمك)
- فقرة <p> فيها وصف بسيط عنك



Day 2: HTML Famous Elements - Forms, Tables, Lists

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>My First Page</title>
</head>
<body>
<form>
    <label>Username:</label>
    <input type="text" name="username" />
    <br />

    <label>Password:</label>
    <input type="password" name="password" />
    <br />

    <input type="submit" value="Login" />
</form>
</body>
</html>
```

في الصفحات التفاعلية، نستخدم 3 عناصر رئيسية باستمرار:

- Forms ✓: لجمع بيانات المستخدم (زي تسجيل الدخول أو إرسال رسالة).
- Tables ✓: لعرض بيانات منظمة (زي جداول الطلاب أو المنتجات).
- Lists ✓: لعرض نقاط أو خطوات (مرتبة أو غير مرتبة).

قم بتجربة وضع هذه العناصر داخل الـ **body**

```
<table border="1">
<tr>
    <th>Name</th>
    <th>Grade</th>
</tr>
<tr>
    <td>Ahmed</td>
    <td>90</td>
</tr>
</table>
```

```
<ol>
<li>Download Editor</li>
<li>Write Code</li>
<li>Save & Run</li>
</ol>
```

```
<ul>
<li>Download Editor</li>
<li>Write Code</li>
<li>Save & Run</li>
</ul>
```



Day 2: Task 2 (3 minutes max)



Task 2 (3 minutes max)

أنشئ صفحة HTML تحتوي على:

- نموذج تسجيل فيه (اسم، بريد، كلمة سر).
- جدول فيه 3 صفوف لأسماء طلاب ودرجاتهم.
- قائمة مرتبة فيها خطوات تعلم البرمجة من وجهة نظرك.



Day 2: HTML Famous Elements - Forms, Tables, Lists

```
<form action="submit.php" method="post">
<label for="name">Full Name:</label>
<input type="text" id="name" name="fullname" />

<label for="email">Email:</label>
<input type="email" id="email" name="email" />

<label for="gender">Gender:</label>
<select id="gender" name="gender">
  <option value="male">Male</option>
  <option value="female">Female</option>
</select>

<label for="message">Message:</label>
<textarea id="message" name="message" rows="4" cols="30"></textarea>

<button type="submit" >send data</button>
</form>
```

أولاً : **Forms** النماذج
الفورم هي الوسيلة اللي بنجمع فيها بيانات من المستخدم.
يتكون من عناصر إدخال زي **input, textarea, select**,
وأزرار إرسال.

- لازم نحدد:
- **action** : فين البيانات هترووح مثلاً: submit.php
 - **method** : طريقة الإرسال GET أو POST

الـ **inputs** لها خواص ف الـ **css** من ضمنها
display: inline
الخاصية دي بتخليل كل عناصر الـ **form** في نفس السطر مع باقي
العناصر وعشان الغيها بروح ل مكان كود الـ **css** وبكتب
input , textarea , button{
 display:block;
}



Day 2: HTML Famous Elements - Forms, Tables, Lists



```
<form  
    action="submit.php"  
    method="post"  
    class="log-form">  
  
    <!-- start the first input with its label -->  
    <label for="name">Full Name:</label>  
    <input type="text" id="name" name="fullname" />  
    <!-- end the first input with its label -->  
  
    <!-- start the second input with its label -->  
    <label for="email">Email:</label>  
    <input type="email" id="email" name="email" />  
    <!-- end the second input with its label -->  
  
    <!-- start the select with its label -->  
    <label for="gender">Gender:</label>  
    <select id="gender" name="gender">  
        <option value="male">Male</option>  
        <option value="female">Female</option>  
    </select>  
    <!-- end the select input with its label -->  
  
    <!-- start the textarea with its label -->  
    <label for="message">Message:</label>  
    <textarea  
        id="message"  
        name="message"  
        rows="4"  
        cols="30"></textarea>  
    <!-- end the textarea with its label -->  
  
    <button type="submit" >send data</button>  
</form>
```

Add some colors to it

```
<style>  
    *{  
        box-sizing: border-box;  
    }  
  
    body {  
        background: linear-gradient(45deg , #3f0 , #af0);  
        min-height: 100vh;  
        padding: 50px 0;  
        margin: 0;  
    }  
  
    input , textarea , button , select {  
        display: block;  
    }  
  
.log-form{  
    width: fit-content;  
    margin: 0 auto;  
    padding: 50px ;  
    background-color: #f2f2f2;  
}  
  
.log-form input , .log-form textarea , .log-form button , .log-form select{  
    width: 100%;  
    margin: 10px;  
    padding: 4px;  
}  
</style>
```



Day 2: HTML Famous Elements - Forms, Tables, Lists

```
<table border="1">
  <thead>
    <tr>
      <th>#</th>
      <th>Name</th>
      <th>Subject</th>
      <th>Grade</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>1</td>
      <td>Ahmed</td>
      <td>Math</td>
      <td>95</td>
    </tr>
    <tr>
      <td>2</td>
      <td>Salma</td>
      <td>Science</td>
      <td>88</td>
    </tr>
    <tr>
      <td>3</td>
      <td>sami</td>
      <td>geography</td>
      <td>75</td>
    </tr>
  </tbody>
</table>
```

```
<table class="log-table" border="1">
  <thead>
    <tr>
      <th>#</th>
      <th>Name</th>
      <th>Subject</th>
      <th>Grade</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>1</td>
      <td>Ahmed</td>
      <td>Math</td>
      <td>95</td>
    </tr>
    <tr>
      <td>2</td>
      <td>Salma</td>
      <td>Science</td>
      <td>88</td>
    </tr>
    <tr>
      <td>3</td>
      <td>sami</td>
      <td>geography</td>
      <td>75</td>
    </tr>
  </tbody>
</table>
```

```
<style>
  *{
    box-sizing: border-box;
  }

  body {
    background: linear-gradient(45deg , #3f0 , #af0);
    min-height: 100vh;
    padding: 50px 0;
    margin: 0;
  }

  .log-table{
    width: fit-content;
    margin: 0 auto;
    padding: 50px ;
    background-color: #f2f2f2;
  }

  .log-table tr td, .log-table tr th {
    padding: 10px;
  }
</style>
```

ثانياً – الجداول
الجداول بتساعدنا نعرض بيانات بشكل منظم.
العنصر الرئيسي هو `<table>` ، وكل
`<tr>` صفت كل خلية `<td>` أو `<th>` للرأس.

Table data
Table head
Table row

نقدر نضيف `thead` , `tbody` , `tfoot` بس منغير هم الجدول هيكون شغال
في تاج `th` بيكون في خصائص تلقائية وهي ان الخط بيكون **bold** بشكل تلقائي لانه عنوان وبيتم وضع `th`
داخل `tr` ك بديل ل `td` ويفضل يكون جوا ال `thead`



Day 2: HTML Famous Elements - Forms, Tables, Lists

```
<h3>Frontend Technologies to Learn:</h3>
<ol>
  <li>HTML</li>
  <li>CSS</li>
  <li>JavaScript</li>
</ol>
```

```
<h3>Shopping List:</h3>
<ul>
  <li>Milk</li>
  <li>Bread</li>
  <li>Eggs</li>
</ul>
```

```
<h3>Frontend Technologies to Learn:</h3>
<ol class="first-ol">
  <li>HTML</li>
  <li>CSS</li>
  <li>JavaScript</li>
</ol>

<ol class="second-ol">
  <li>HTML</li>
  <li>CSS</li>
  <li>JavaScript</li>
</ol>

<h3>Shopping List:</h3>
<ul class="first-ul">
  <li>Milk</li>
  <li>Bread</li>
  <li>Eggs</li>
</ul>

<ul class="second-ul">
  <li>Milk</li>
  <li>Bread</li>
  <li>Eggs</li>
</ul>
```

```
<style>
  *{
    box-sizing: border-box;
  }

  body {
    background: linear-gradient(45deg , #3f0 , #af0);
    min-height: 100vh;
    padding: 50px 10px;
    margin: 0;
    color: #333
  }

  .first-ul{
    list-style-type : georgian;
  }

  .first-ol {
    list-style-type: lower-alpha| ;
  }
</style>
```

ثالثاً – القوائم
القوائم بتحلي المحتوى
مقروء أكثر ومرتب.
فيه نوعين:
 • قائمة مرتبة
 • قائمة غير مرتبة

العناصر المرتبة نقدر تغيير طريقة الترتيب 1 2 3 ل مثلا a b c او لاتيني وهكذا
العناصر الغير مرتبة نقدر تغيير في ال icon جنب كل عنصر في القائمة باتاعتها

كل عنصر داخل القائمة هو



Day 2: Task 3 (3 minutes max)

Task 3 (3 minutes max)

أنشئ صفحة HTML تحتوي على:

- Form لتسجيل مستخدم جديد (اسم - بريد - كلمة سر - اختيار الجنس – (textarea
- جدول يحتوي على 5 طلاب، كل صف فيه:
رقم - اسم - القسم - الدرجة.
- قائمتين:

واحدة مرتبة فيها خطوات مشروعك الأول،

وآخرى غير مرتبة فيها الأدوات اللي بتسخدمها.

غير الـ `list-style-type` في القائمتين المرتبة خليها بالحروف وغير مرتبة خليها باي شكل اخر

1 Syntax to write CSS

Selectors

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

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

Property and its value

This is the actual style to be applied to the element(s)

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

elementname

.classname

#idname

[attr=value]

*

Combinators

selectorA + selectorB

Adjacent sibling

selectorA ~ selectorB

General sibling

parent > child

Direct child

parent descendant

Descendent

Pseudo Selectors

:active

:hover

:visited

: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

margin

border

padding
content

BOX:

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

POSITION:

position
top
bottom
left
right
z-index



Day 2: CSS intro (Cascading Style Sheets)



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

elementname

.classname

#idname

[attr=value]

Combinators

selectorA + selectorB Adjacent sibling

selectorA ~ selectorB General sibling

parent > child Direct child

parent descendant Descendent

Pseudo Selectors

:active

:hover

:visited

:focus



(Fonts)

الخط نوع تحديد: font-family:
الخط حجم تحديد: font-size:
الخط سمك تحديد: font-weight:
الخط نمط تحديد: font-style:
line-height: بين السطر ارتفاع تحديد
الأسطر.

01

02

03

04

05

06

(Color)

النص لون تحديد: color:
لون تحديد: background-color:
الخلفية: opacity:
العنصر شفافية درجة تحديد: border-color:
الحدود لون تحديد:

(layout)

العنصر عرض طريقة تحديد: display:
بالنسبة للعنصر موقع تحديد position:
(static, relative,
absolute, fixed).
المسافة تحديد: top, right, bottom, left:
استخدام عند الحواف من position.

(Borders)

الحدود ولون ونوع سمك تحديد: border:
المدورة الزوايا تحديد: border-radius:
للسعر.
الحدود نوع تحديد: border-style:
الحدود سمك تحديد: border-width:

(Spacing)

وحدود المحتوى بين المسافة: padding:
العنصر.
والعناصر العنصر بين المسافة: margin:
الأخرى.
الكلي الحجم حساب طريقة تحديد: box-sizing:
للسعر.

(Advanced Styles)

حول ظل إضافة: box-shadow:
العنصر.
للنص ظل إضافة: text-shadow:
على بصرية تأثيرات تطبيق: filter:
التبابين أو التشويف مثل العنصر.

ممكن اكتب كود css

في نفس السطر مع كود الـ html

inline

في نفس الصفحة مع كود الـ html

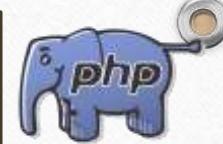
internal

خارج الصفحة واستدعيه في كود الـ html

external



Day 2: CSS intro (Cascading Style Sheets)



```
<style>
  *{
    box-sizing: border-box;
  }

  body {
    font-family: Arial, sans-serif;
    background-color: #f2f2f2;
    padding: 20px;
  }

  form {
    background: white;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 0 10px #ccc;
  }

  input,
  textarea,
  select,
  button {
    width: 100%;
    padding: 10px;
    margin-top: 8px;
    border: 1px solid #ccc;
    border-radius: 5px;
  }
</style>
```

```
table {
  width: 100%;
  border-collapse: collapse;
  margin-top: 20px;
}

table th {
  background: #007bff;
  color: white;
  padding: 10px;
}

table td {
  padding: 10px;
  text-align: center;
  border: 1px solid #ccc;
}

ul,
ol {
  background: #fff;
  padding: 15px 15px 15px
  50px;
  border-radius: 5px;
}
</style>
```

Cascading Style Sheets معناها "CSS"
هي اللغة المسؤولة عن الشكل (الستايل) داخل صفحات
الويب:
الوان، حدود، هوامش، خطوط، أحجام... الخ.

selector {
 property: value;
}

For example:-

```
h1{
  color: green;
}
```

مثلاً:
element →
p, table, form

.class →
.class
نستهدف عناصر معينة

#id →
#id
نستهدف عنصر افرد عند ID

كل العناصر → *



Day 2: CSS intro (Cascading Style Sheets)



```
<style>
  *{
    box-sizing: border-box;
  }

  body {
    font-family: Arial, sans-serif;
    background-color: #f2f2f2;
    padding: 20px;
  }

  form {
    background: white;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 0 10px #ccc;
  }

  input,
  textarea,
  select,
  button {
    width: 100%;
    padding: 10px;
    margin-top: 8px;
    border: 1px solid #ccc;
    border-radius: 5px;
  }

  table {
    width: 100%;
    border-collapse: collapse;
    margin-top: 20px;
  }
```

```
  table th {
    background: #007bff;
    color: white;
    padding: 10px;
  }

  table td {
    padding: 10px;
    text-align: center;
    border: 1px solid #ccc;
  }

  ul,
  ol {
    background: #fff;
    padding: 15px 15px 15px 50px;
    border-radius: 5px;
  }
</style>
</head>
<body>
```

```
<form
  action="submit.php"
  method="post"
  class="log-form">

  <!-- start the first input with its label -->
  <label for="name">Full Name:</label>
  <input type="text" id="name" name="fullname" />
  <!-- end the first input with its label -->

  <!-- start the second input with its label -->
  <label for="email">Email:</label>
  <input type="email" id="email" name="email" />
  <!-- end the second input with its label -->

  <!-- start the select with its label -->
  <label for="gender">Gender:</label>
  <select id="gender" name="gender">
    <option value="male">Male</option>
    <option value="female">Female</option>
  </select>
  <!-- end the select input with its label -->

  <!-- start the textarea with its label -->
  <label for="message">Message:</label>
  <textarea
    id="message"
    name="message"
    rows="4"
    cols="30"></textarea>
  <!-- end the textarea with its label -->

</form>
```

```
<h3>Frontend Technologies to Learn:</h3>
<ol class="first-ol">
  <li>HTML</li>
  <li>CSS</li>
  <li>JavaScript</li>
</ol>

<ol class="second-ol">
  <li>HTML</li>
  <li>CSS</li>
  <li>JavaScript</li>
</ol>

<h3>Shopping List:</h3>
<ul class="first-ul">
  <li>Milk</li>
  <li>Bread</li>
  <li>Eggs</li>
</ul>

<ul class="second-ul">
  <li>Milk</li>
  <li>Bread</li>
  <li>Eggs</li>
</ul>
```





Day 2: CSS intro (Cascading Style Sheets)



```
<table class="log-table" border="1">
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Subject</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ahmed</td>
<td>Math</td>
<td>95</td>
</tr>
<tr>
<td>2</td>
<td>Salma</td>
<td>Science</td>
<td>88</td>
</tr>
<tr>
<td>3</td>
<td>sami</td>
<td>geography</td>
<td>75</td>
</tr>
</tbody>
</table>
```

Final output

Full Name: _____

Email: _____

Gender: Male Female

Message:

send date

Frontend Technologies to Learn:

- 1. HTML
- 2. CSS
- 3. JavaScript

Shopping List:

- Milk
- Bread
- Eggs

• Milk

• Bread

• Eggs

#	Name	Subject	Grade
1	Ahmed	Math	95
2	Salma	Science	88
3	sami	geography	75



Day 2: Task 4 (3 minutes max)



Task 4 (3 minutes max)

مطلوب منكم:

إعادة استخدام نفس عناصر HTML (form + table + list).

لكن تعمدوا تناسق مختلف تماماً:

✓ ألوان مختلفة

✓ خطوط مختلفة

✓ خلفيات مختلفة

✓ استخدام `id`# و `class`.



Day 2: CSS more info



```
<div class="card center">
  <h2>Submit Form</h2>
  <button class="btn">Send</button>
</div>
```

```
<style>
.card {
  background-color: white;
  padding: 20px;
  margin: 20px auto;
  border-radius: 10px;
  box-shadow: 0 2px 8px rgba(0,0,0,0.1);
  width: 80%;
}
.center {
  text-align: center;
}
```

```
.btn {
  background-color: #007bff;
  color: white;
  padding: 10px 20px;
  border: none;
  border-radius: 6px;
  cursor: pointer;
}

.btn:hover {
  background-color: #0056b3;
}

</style>
```

دلوقي بعد ما فهمنا الشكل العام للـ **CSS**، هنبدأ نطبق أكثر
بواقعية:
إزاي نستخدم **class** لتكرار التنسيق على أكثر من عنصر؟
إزاي نستخدم **hover** علشان نغير لون الزر لما الماوس يقف
عليه؟
إزاي نستخدم **border- radius** علشان نخلي الشكل أنعم؟
padding و **margin** علشان نخلي الشكل أنعم؟

ستابل كارت متكرر ، ممكن نستخدمه مع كل عناصر **card**:
الصفحة.
زر أنيق وثابت الشكل.
دا اسمه **pseudo-class hover**، بيخلify العنصر يتغير
بمجرد ما يحصل حدث معين (زي وقوف الماوس).
استخدمنا **box-shadow** علشان ندي عمق للعناصر.



Day 2: CSS more info



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        .card {
            background-color: #fff;
            padding: 20px;
            margin: 20px auto;
            border-radius: 10px;
            box-shadow: 0 2px 8px rgba(0,0,0,0.1);
            width: 80%;
        }

        .btn {
            background-color: #007bff;
            color: #fff;
            padding: 10px 20px;
            border: none;
            border-radius: 6px;
            cursor: pointer;
        }

        .btn:hover {
            background-color: #0056b3;
        }

        .center {
            text-align: center;
        }
    </style>
</head>
<body>

    <div class="card center">
        <h2>Submit Form</h2>
        <button class="btn">Send</button>
    </div>

</body>
</html>
```

```
<div class="card center">
    <h2>Submit Form</h2>
    <button class="btn">Send</button>
</div>

<div class="card center">
    <h2>Submit Form</h2>
    <button class="btn">Send</button>
</div>

<div class="card center">
    <h2>Submit Form</h2>
    <button class="btn">Send</button>
</div>

<div class="card center">
    <h2>Submit Form</h2>
    <button class="btn">Send</button>
</div>
```

مع تكرار العنصر
تتكرر التنسيقات

Submit Form

Send



Day 2: Task 5 (3 minutes max)

Task 5 (3 minutes max)

صمّم 3 أقسام (div) كل واحد فيهم عبارة عن كارت:

kart فيه form

kart فيه جدول

kart فيه قائمة un ordered

استخدم كلاس card لتنسيقهم بنفس الشكل.

أضف زر أسفل كل كارت عليه .btn وفعّل له تأثير hover لتغيير الخلفية

اضف بعض المرونة في تأثير الـ hover من خلال إضافة

(all .5s ease-in-out) value (transition) property



Day 2: HTML + CSS



```
<nav>
  <ul>
    <li><a href="#">Home</a></li>
    <li><a href="#">About</a></li>
    <li><a href="#">Courses</a></li>
    <li><a href="#">Contact</a></li>
  </ul>
</nav>
```

```
<style>
  nav {
    background-color: #333;
    padding: 15px;
    border-radius: 8px;
  }
```

```
nav ul {
  list-style: none;
  display: flex;
  gap: 20px;
  justify-content: center;
  margin: 0;
  padding: 0;
}
```

```
nav ul li a {
  color: white;
  text-decoration: none;
  font-weight: bold;
  padding: 8px 12px;
  border-radius: 4px;
  transition: background 0.3s;
}
```

```
nav ul li a:hover {
  background-color: #555;
}
</style>
```

استخدمنا **display: flex** علشان نخلی القوائم أفقية.
بتتنظم المسافة بينهم.
gap بتخلی التنقل تفاعلي.
استخدمنا **background-color** و **border-radius** بيدوا لمسة جمالية.

عنصر **<nav>** هو المكان اللي بنحط فيه روابط التنقل الأساسية في الموقع:
زي: Home – About – Contact – Login
بنستخدم داخله روابط **<a>** وأحياناً **** علشان نعرضهم بشكل أفقي ومنسق.
وهننسقه بـ CSS علشان بيان زي Navbar حقيقي.

ستایل کارت متکرر، ممکن نستخده مع کل عناصر **card**:
الصفحة.
زر انيق وثابت الشكل.
btn.
دا اسمه **pseudo-class hover**: بمجرد ما يحصل حدث معين (زي وقوف الماوس).
استخدمنا **box-shadow** علشان ندي عمق للعناصر.



Day 2: 📝 Task 6 (from home)



📍 Task 6 (from home)

أنشئ شريط تنقل خاص بيك فيه:

على الأقل 4 روابط

كل رابط يروح لمكان وهمي (#)

استخدم ألوان وأحجام مختلفة عن المثال

لازم التنقل يبقى أفقى

❸ أضف لمسة hover خاصة بيك (لون، حجم، ظل... إلخ)

task

خاصية ال shadow

box-shadow: *h-offset v-offset blur spread color*;

```
.log-form{  
    width: fit-content;  
    margin: 30px auto;  
    padding: 50px ;  
    background-color: #f2f2f2;  
    box-shadow: 0 10px 20px rgba(0,0,0,0.1);  
    font-family: "comic sans ms";  
}
```

Full Name:

Email:

Gender:

Male

Message:

Send Data



Day 2: Project Files arrangements

ترتيب الملفات والمجلدات الخاص بك جزاً مهم جداً من المشروع

مثال على استدعاء ملف CSS خارجي

استدعى الملفات التي تحتاجها في الكود بنظام

Name	Date modified	Type	Size
css	٢٠٢٤/٠١/٠٣ ٩:٥٠	File folder	
js	٢٠٢٤/٠١/٠٣ ٩:٥٠	File folder	
index.html	٢٠٢٤/٠١/٠٣ ١٠:٩٩	Microsoft Edge HT...	3 KB

```
1 <!doctype html>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1">
6     <title>Bootstrap demo</title>
7     <link href="css/bootstrap.min.css" rel="stylesheet" >
8     <link rel="stylesheet" href="css/style.css">
9   </head>
10  <body>
```

```
48
49    <script src="js/bootstrap.bundle.min.js"></script>
50    <script src="js/popper.min.js"></script>
51    <script src="js/bootstrap.min.js"></script>
52    <script src="js/script.js"></script>
53  </body>
54  </html>
```

Day 2: Js (javaScript)

- "JavaScript" هي لغة البرمجة اللي بتخلي الصفحة تتحرك وتفاعل مع المستخدم.
- بتشتغل من جهة المستخدم (Client-side) ، وبتسمح لنا نتحكم في المحتوى، نرد على الأحداث (زي الضغط على زر)، ونتعامل مع البيانات.
- JavaScript بتكتب داخل الصفحة باستخدام <script>، ويفضل دايماً تكتبها في آخر الصفحة قبل </body>. كيف يمكنني استخدام لغة الjs وماذا يمكنها ان تفعل
- طريقة كتابة اللغة (Tags) Syntax (Tags)
- المتغيرات والبيانات و انواعها Variables / Constants & datatypes



Day 2: JS intro



```
<!DOCTYPE html>
<html>
<head>
  <title>JS Intro</title>
</head>
<body>

<h2 id="greeting">Hello!</h2>
<button onclick="sayWelcome()">Click Me</button>

<script>
  function sayWelcome() {
    document.getElementById("greeting").innerHTML = "Welcome to JavaScript!";
  }
</script>

</body>
</html>
```

- ✓ عرفنا دالة () اسمها `sayWelcome()`.
- ✓ ربطنا الزر بـ `onclick` علشان لما المستخدم يضغط،
الكود ينتفذ.
- ✓ استخدمنا `document.getElementById()` علشان
نوصل لعنصر ونغير محتواه.



Day 2: Task 5 (3 minutes max)

Task 5 (3 minutes max)

أنشئ صفحة فيها:

- عنوان `<h1>` فيه كلمة "مرحبا"
- زر `<button>` مكتوب عليه "غير العنوان"
- عند الضغط على الزر، يتغير العنوان إلى "أهلاً بك في الجافاسكريبت"
- أضف زر ثاني يغير لون الخلفية لما المستخدم يضغط عليه.



Day 2: JS intro



```
<!DOCTYPE html>
<html>
<head>
  <title>JS Input Example</title>
</head>
<body>

<h2>Say Hello</h2>

<input type="text" id="username" placeholder="Enter your name" />
<button onclick="sayHello()">Say Hello</button>
```

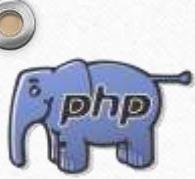
```
<p id="output"></p>

<script>
  function sayHello() {
    let name = document.getElementById("username").value;
    document.getElementById("output").innerHTML = "Hello, " + name + "!";
  }
</script>

</body>
</html>
```

✓ دلوقتي هنتعلم إزاي نقرأ البيانات اللي المستخدم بيكتبها
في **<input>** ونتفاعل معها باستخدام الأحداث
click, change, keyup. (Events)
أهم جزء هنا هو: ✓
نستخدم **document.getElementById().value**
عشان ناخذ القيمة اللي المستخدم دخلها.

معناها إحنا أخذنا القيمة اللي المستخدم ... = كتبها.
بعدها استخدمنا **innerHTML** عشان نطبع الكلام. ✓
ممكن نستخدم أي حدث تاني زي **onchange, onkeyup** ✓



Day 2: Task 6 (5 minutes max)



Task 6 (5 minutes max)

صمّم صفحة فيها:

حقل إدخال لاسم الطالب

زر مكتوب عليه "أظهر النتيجة"

أضف حدث `oninput` لعنصر ادخال الاسم يظهر كتابة المستخدم لحظياً.

عند الضغط على الزر، يظهر تحت الزر:

"مرحباً يا [اسم]" ← يتم استبدال الاسم بناءً على ما كتب ، "نتيجة = 90%" .



Day 2: JS conditions



```
<input type="number" id="score" placeholder="Enter your score" />
<button onclick="checkResult()">Check Result</button>
<p id="result"></p>

<script>
function checkResult() {
    let score = document.getElementById("score").value;
    if (score >= 50) {
        document.getElementById("result").innerHTML = " ✅ ";
    } else {
        document.getElementById("result").innerHTML = " ❌ ";
    }
}
</script>
```

✓ الشرط `if` في JavaScript يبخلينا ننفذ كود معين لما شرط معين يتحقق.

✓ هو من أهم أدوات اتخاذ القرار داخل الكود.

✓ الشكل العام:

```
if(شرط) {
    الكود اللي يتنفذ لو الشرط اتحقق //
} else {
    الكود اللي يتنفذ لو الشرط ما اتحقق //
}
```

✓ استخدمنا `(score >= 50)` `if` للتحقق من النجاح.

✓ لو الشرط اتحقق يطبع "ناجح"، لو لا يطبع "راسب".

✓ ممكن نستخدم `else if` كمان لو عندنا أكثر من شرط.

```
if (score >= 90) {
    document.getElementById("result").innerHTML = "GPA (A)";
} else if (score >= 80) {
    document.getElementById("result").innerHTML = "GPA (B)"
} else if (score >= 70) {
    document.getElementById("result").innerHTML = "GPA (C)"
} else {
    document.getElementById("result").innerHTML = "GPA (F)"
}
```



Day 2: Task 6 (5 minutes max)



Task 6 (5 minutes max)

أنشئ صفحة فيها:

- حقل إدخال للدرجات (من 0 إلى 100)
- اكتب اسمك في عنوان `h1`
- زر "احسب التقدير"
- عند الضغط، يظهر التقدير (ممتاز - جيد جداً - جيد - ضعيف)
- استخدم `if`, `else if`, `else` بالشكل المناسب.
- (بونص) قم بطباعة النتيجة داخل `div` به تنسيقات `CSS` مثل خلفية ولون نص واي تنسيقات إضافية لك



Day 2: JS Loops



```
<button onclick="printNumbers()">Print 1 to 5</button>
<ul id="numList"></ul>

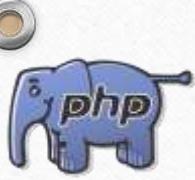
<script>
function printNumbers() {
    let output = "";
    for (let i = 1; i <= 5; i++) {
        output += "<li>Number " + i + "</li>";
    }
    document.getElementById("numList").innerHTML = output;
}
</script>
```

- ✓ التكرار (loop) بيسمح لنا نكرر تنفيذ كود معين أكثر من مرة بشكل تلقائي.
- ✓ مفید جداً لما نحب نطبع عناصر كثير أو نعد أرقام أو نكرر فحص بيانات.
- ✓ أول نوع هنتعلم هو: `for` loop ← وده بيشتغل بعدد محدد من المرات.

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

- ✓ بيبدا من `i = 0`
- ✓ ينفذ الكود طالما `i < 5`
- ✓ ويزيّد `i` في كل مرة بمقدار `1 (i++)`
- ✓ (`i++ → i+=1 → i = i + 1`)

استخدمنا `for` علشان نطبع 5 أرقام في `` استخدمنا `innerHTML` علشان نحدث الصفحة. ممكن نغير العدد أو الشرط حسب الحاجة.



Day 2: Task 6 (5 minutes max)



Task 6 (5 minutes max)

صمّم صفحة فيها:

- زر مكتوب عليه "طباعة من 1 لـ 10" عند الضغط، تطبع الأرقام من 1 لـ 10 بدلاً من اطبعها كأزرار أو جوا جدول.

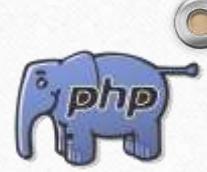


Break

Break



Day 2: 📄 Task 7 (15 minutes max)



💡 Task 6 (15 minutes max)

A screenshot of a web browser window showing a "Student Registration" form. The URL in the address bar is "localhost/PHP_training/Day2/basics/project.html". The form has three input fields: "Name" (ahmed), "Email" (ahmedabubakr148@gmail.com), and "Level" (First Year). A blue "Register" button is below the fields. A green success message box at the bottom left displays "✓ Registration Complete!" followed by the submitted data: Name: ahmed, Email: ahmedabubakr148@gmail.com, and Level: First Year.

إنشاء نموذج بسيط فيه حقول لإدخال البيانات
(اسم - بريد إلكتروني - سنة دراسية)، وبعد
الضغط على الزر، يتم عرض البيانات المدخلة
بشكل أنيق على الصفحة.

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() { }

2 Basic Vocabulary

Variable

A named reference to a value is a variable.

`var a = 7 + "2";`

Operator

Operators are reserved-words that perform action on values and variables.
Examples: + - = * in === typeof != ...

Statement

A group of words, numbers and operators that do a task is a statement.

Note: var, let & const are all valid keywords to declare variables. The difference between them is covered on page 7 of this cheatsheet.

Keyword / reserved word
Any word that is part of the vocabulary of the programming language is called a keyword (a.k.a reserved word). Examples: var = + if for...

Expression
A reference, value or a group of reference(s) and value(s) combined with operator(s), which result in a single value.

The screenshot shows a browser's developer tools console window. The top bar includes icons for back/forward, top/filter, and settings, along with a message count of 5. The console itself displays the following JavaScript session:

```
> var x = 1;
< undefined
> var y = 2
< undefined
> var z
< undefined
> z = x + y
< 3
> var name = "ahmed"
< undefined
> console.log("hello" + name)
helloahmed
< undefined
> console.log("hello " + name)
hello ahmed
< undefined
> alert("hello " + name)
< undefined
>
```

On the left side of the console, there is a vertical toolbar with various icons: a square, a house, a double slash, a magnifying glass, a signal, a gear, a square with a plus, and a plus sign.



Day 2: JS intro



The screenshot shows a code editor with three examples of JavaScript output:

```
<script>
  console.log("this is printed in console");
  alert("this is prented on alert");
  document.write("this is printed in html file")
</script>
```

The code editor has three panels:

- Top Panel:** Shows the code above. A callout box highlights "this is printed in console".
- Middle Panel:** Shows the browser developer tools' Console tab. It displays the message "this is printed in console". A callout box highlights this message.
- Bottom Panel:** Shows the browser developer tools' Elements tab. It displays the message "this is printed in html file" within the body element. An arrow points from the message in the console to the corresponding line in the DOM tree.

The browser tabs at the top show "File D:/session-8-2-2025/Day%203/proj/index.html".

5 Vocabulary around variables and scope

```
var a;
```

Variable Declaration

The creation of the variable.

```
a = 12;
```

Variable Initialization

The initial assignment of value to a variable.

```
a = "me";
```

Variable Assignment

Assigning value to a variable.

```
console.log(a);  
var a = "me";
```

Hoisting

Variables are declared at the top of the function automatically, and initialized at the time they are run.

Scope

The limits in which a variable exists.

Global scope

The outer most scope is called the Global scope.

Functional scope

Any variables inside a function is in scope of the function.

Lexical Environment (Lexical scope)

The physical location (scope) where a variable or function is declared is its lexical environment (lexical scope).

Rule:

(1) Variables in the outer scope can be accessed in a nested scope; But variables inside a nested scope CANNOT be accessed by the outer scope. (a.k.a private variables.)

(2) Variables are picked up from the lexical environment.

```
var a = "global";
```

```
function first(){
```

```
    var a = "fresh";
```

```
    function second(){
```

```
        console.log(a);
```

```
}
```

```
}
```

Scope chain

The nested hierarchy of scope is called the scope chain. The JS engine looks for variables in the scope chain upwards (it its ancestors, until found)

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
```

تاج مجمع لنتائج معاه

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.

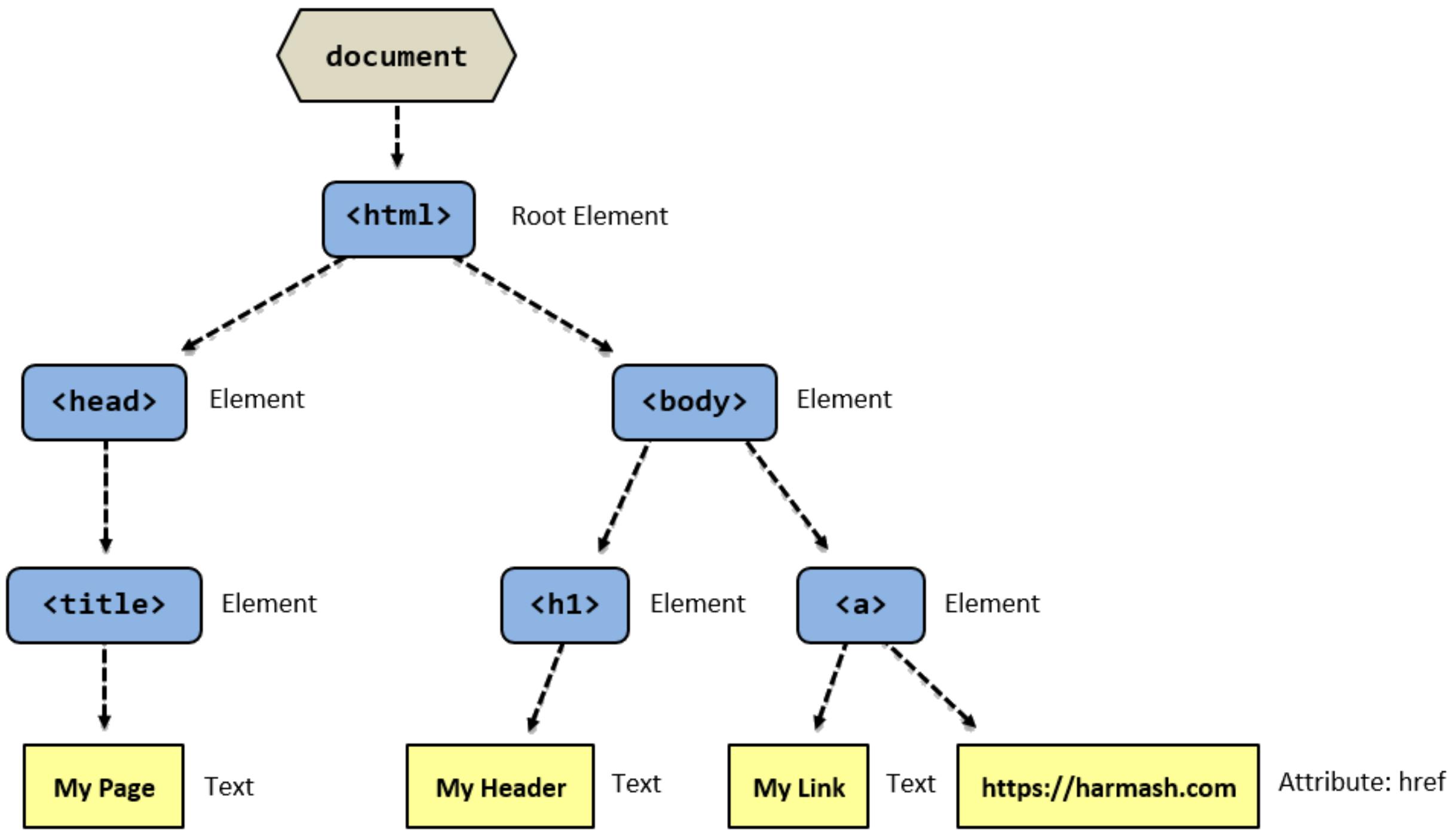
Element Node: Node that has an element.

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.



A screenshot of a web developer's browser interface showing the Elements, Console, and Scripts panels.

Elements Panel: Shows the HTML structure of the page. A script block is selected, containing the following code:

```
var dataHeader = document.getElementById("theLabel").innerHTML;  
console.log(dataHeader);
```

The element `<h1 id="theLabel">sign-up for` is highlighted with a red box and has a yellow arrow pointing from it to the **Console** tab.

Console Tab: Displays the output of the selected script: `sign-up form`. The output is timestamped as `index.html:20`.

Script Panel: Shows the full script content:

```
<script>  
var dataHeader =  
document.getElementById("theLabel").innerHT  
console.log(dataHeader);  
</script>
```

Right Panel: Shows a visual representation of the sign-up form. It includes four input fields: "First name", "Last name", "email", and "password", and a "Login!" button.

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

→ Relational / Comparison

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

→ 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

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/operatorstable>

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  
}
```

10 Loop Statements

Loops are used to do something repeatedly. For instance lets say we get a list of 50 blog posts from the database and we want to print their titles on our page. Instead of writing the code 50 times, we would instead use a loop to make this happen.

For loop

```
for (initial-expression; condition; second-expression){  
    // run this code in block  
}
```

Step 1: Run the initial expression.

Step 2: Check if condition meets. If condition meets, proceed; or else end the loop.

Step 3: Run the code in block.

Step 4: Run the second-expression.

Step 5: Go to Step 2.

While loop

```
while (i<3){  
    // run this code in block  
    i++;  
}
```

Step 1: If the condition is true, proceed; or else end the loop.

Step 2: Run the code in block.

Step 3: Go to Step 1.

Do while loop

```
do {  
    // run this code in block  
    i++;  
} while (i<3);
```

Step 1: Run the code in block.

Step 2: If the condition is true, proceed; or else end the loop.

Step 3: Go to Step 1.

A function is simply a bunch of code bundled in a section. This bunch of code ONLY runs when the function is called. Functions allow for organizing code into sections and code reusability.

Using a function has ONLY two parts. (1) Declaring/defining a function, and (2) using/running a function.

Name of function

That's it, its just a name you give to your function.

Tip: Make your function names descriptive to what the function does.

Return (optional)

A function can optionally spit-out or "return" a value once its invoked. Once a function returns, no further lines of code within the function run.

Invoke a function

Invoking, calling or running a function all mean the same thing. When we write the function name, in this case `someName`, followed by the brackets symbol () like this `someName()`, the code inside the function gets executed.

```
// Function declaration / Function statement
function someName(param1, param2){
    // bunch of code as needed...
    var a = param1 + "love" + param2;
    return a;
}
```

```
// Invoke (run / call) a function
someName("Me", "You")
```

Parameters / Arguments (optional)

A function can optionally take parameters (a.k.a arguments). The function can then use this information within the code it has.

Code block

Any code within the curly braces {} is called a "block of code", "code block" or simply "block". This concept is not just limited to functions. "if statements", "for loops" and other statements use code blocks as well.

Passing parameter(s) to a function (optional)

At the time of invoking a function, parameter(s) may be passed to the function code.

"Day 2: Summary Slide for front-end – What We Learned "

الهيكل الأساسي لأي صفحة HTML 

تنسيقات CSS أساسية (ألوان - هوامش - حواف - خطوط)

عشنان نقدر نتفاعل مع الصفحة JavaScript

إزاي نقرأ البيانات من المستخدم ونطبعها

استخدام الشروط والتكرار

مشروع صغير بيجمعهم مع بعض



"Day 2: Summary Slide for front-end – What We Learned "

" كل دا بياحسننا إننا نبني صفحات ذكية بتفاعل مع المستخدم. لكن في الشغل الحقيقي، مش بنطبع بيانات كده مباشره! دائمًا في سيرفر وداتا بيز."

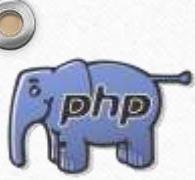
ودا اللي هيقودنا ليوم جديد فيه دمج بين:

HTML + CSS + JS ✓

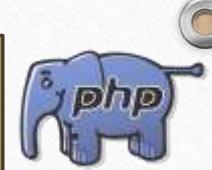
✓ و PHP بلغة backend

✓ مع مفهوم الـ data-bean لاحقاً





Day 2: Task 8 home work



Task 8 home work

صمّم صفحة بسيطة يكون فيها:

نص ترحبي متغير بـ JavaScript

نموذج فيه حقل اسم، وظهور رسالة عند الإدخال

عنصر HTML جديد (مثلاً <table> أو <nav>) من تنسيق

الغرض إنك تربط بين المفاهيم اللي اشتغلنا بيها وتستعد إنك تكتب كود "مترابط".



Day 2: Project Files arrangements

ترتيب الملفات والمجلدات الخاص بك جزاً مهم جداً من المشروع

مثال على استدعاء ملف CSS خارجي

استدعى الملفات التي تحتاجها في الكود بنظام

Name	Date modified	Type	Size
css	٢٠٢٤/٠١/٠٣ ٩:٥٠	File folder	
js	٢٠٢٤/٠١/٠٣ ٩:٥٠	File folder	
index.html	٢٠٢٤/٠١/٠٣ ١٠:٩٩	Microsoft Edge HT...	3 KB

```
1 <!doctype html>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1">
6     <title>Bootstrap demo</title>
7     <link href="css/bootstrap.min.css" rel="stylesheet" >
8     <link rel="stylesheet" href="css/style.css">
9   </head>
10  <body>
```

```
48
49    <script src="js/bootstrap.bundle.min.js"></script>
50    <script src="js/popper.min.js"></script>
51    <script src="js/bootstrap.min.js"></script>
52    <script src="js/script.js"></script>
53  </body>
54  </html>
```

The End Of Day-2



Feel free to contact me any time

linktr.ee/ahmed_abubakr