Final Module
Summary and Cloud Deployment

דגשים לבחינות

TomerBu

4 טפסים

HTML & CSS

JS & React

C# including WPF

ASP.NET & SQL & Entity Framework & NoSQL

HTML & CSS

ראשי תיבות
למה צריך HTML

תפקיד ראשי תיבות וכו'. CSS

תגיות שצריך להכיר:

<!doctype>, html, head, link, script

כולל איך כותבים ומה הסינטקס

Body, h1-h6, p, a, img, table, tr, td, div, span, form, input, button, label, textarea,
- סוגי רשימות
ol, ul, dl

Inline elements vs block elements – display property (inline, block, grid, flex)

HTML + CSS

תגיות ושימושן:

`<html>` : תגית השורש של המסמך שמכילה את כל התוכן של עמוד האינטרנט.

`<head>` : מכילה מידע על המסמך כמו כותרת הדף, קישורים לקבצים חיצוניים (CSS, JavaScript) ומטא-דאטה.

`<title>` : מגדירה את כותרת הדף שמופיעה בלשונית הדפדפן.

`<body>` : מכילה את כל התוכן הנראה בעמוד, כולל טקסט, תמונות, קישורים, וטפסים.

`<h1>` **עד** `<h6>` : כותרות ברמות שונות. `<h1>` היא הכותרת הראשית, ו- `<h6>` היא כותרת בדרגה נמוכה יותר.

`<p>` : פסקה של טקסט.

`<a>` : קישור המוביל לעמוד אחר או לנקודה מסוימת באותו עמוד.

`<img>` : מציגה תמונה.

`<li>` , `<ol>` , `<ul>` : רשימות. `<ul>` היא רשימה לא מסודרת, `<ol>` היא רשימה מסודרת, ו- `<li>` מייצגת פריט ברשימה.

`<div>` : תגית כללית המשמשת לארגון קבוצות של אלמנטים או לעיצוב מבני, ללא משמעות סמנטית.

`<span>` : תגית פנימית שמסמנת חלק מטקסט או קבוצה קטנה של אלמנטים, גם היא ללא משמעות סמנטית.

`<form>` : מגדירה טופס קלט למילוי נתונים. בדרך כלל משמשת יחד עם:

`<input>` : שדה קלט בסיסי.

`<button>` : כפתור לביצוע פעולה כמו שליחת טופס.

`<label>` : תווית עבור שדה קלט, כדי להקל על זיהוי התוכן שיש למלא.

`<textarea>` : אזור טקסט רחב שמאפשר הכנסת כמות גדולה של טקסט.

`<th>` , `<td>` , `<tr>` , `<table>` : טבלה. `<table>` מגדיר את הטבלה, `<tr>` מגדיר שורה, `<td>` מגדיר תא, ו- `<th>` מגדיר כותרת לטבלה.

HTML מיועדת להגדרת תוכן העמוד
CSS לעיצוב העמוד

HTML = Hyper text markup language

CSS= Cascading style sheets

HTML + CSS

תגיות ושימושן:

תגיות סמנטיות:

CSS Semantic Tags

- `<header>` : מייצגת תוכן מבואי או קבוצת קישורים לניווט. בדרך כלל כוללת את הלוגו, תפריט הניווט, וכו'.
- `<nav>` : מגדירה קונטיינר לקישורי ניווט.
- `<section>` : מייצגת חלק כללי במסמך, משמשת לקיבוץ תוכן עם נושא משותף.
- `<article>` : מכילה תוכן עצמאי שאפשר לפרסם בנפרד, כמו פוסט בבלוג או כתבת חדשות.
- `<aside>` : מייצגת תוכן שאינו בהכרח קשור לתוכן המרכזי – לכן הוא מכיל פרסומות, קישורים חיצוניים, או תוכן משלים אחר.
- `<footer>` : מכילה מידע תחתון לסעיף או לכל הדף, כמו זכויות יוצרים או קישורים נוספים.
- `<main>` : מייצגת את התוכן המרכזי של הדף, שייחודי לכל עמוד, ולא כוללת כותרות, כותרות תחתונות או סרגלי צד.
- `<figure>` ו- `<figcaption>` : משמשות להצגת תמונות, תרשימים או איורים עם כיתוב אופציונלי (`figcaption`).

שימוש בתגיות אלה משפר את קריאות הקוד ומסייע למנועי חיפוש להבין טוב יותר את מבנה הדף.

התגיות משמשות את המפתחים לכתיבה סמנטית, את הדפדפנים להבנה העמוד ולרינדור שלו, את מנועי החיפוש להבנת העמוד, ואת כלי ההנגשה להבנת העמוד.

עוד תגיות חשובות:

header, nav, footer, article, section, main, aside, figure, figcaption

להכיר את השימוש ולמה משתמשים בכל תגית.

תגיות נוספות:
<link>, <img>, <a>, <script>

לדעת את הסינטקסט של התגיות האלה

HTML + CSS

תגיות ושימושן:

תגית קישור:

כך מגדירים קישור ב-HTML:

```
<a/>לחץ כאן למעבר לאתר"a href="https://example.com>
```

`href` מציין את כתובת ה-URL שאליה הקישור יוביל.

הטקסט בתוך התגית ("לחץ כאן למעבר לאתר") הוא מה שיוצג למשתמש כלחיץ.

תגית תמונה:

כך מגדירים תמונה ב-HTML:

```
<"תיאור התמונה"=img src="path/to/image.jpg" alt
```

`src` מציין את הנתיב של קובץ התמונה.

`alt` מספק תיאור טקסטואלי לתמונה במקרה שהיא לא נטענת או למטרות נגישות.

אין צורך בסגירה של `<img>` , מאחר שהיא תגית ריקה (היא לא עוטפת תוכן נוסף).

כדי להוסיף הערה, יש להשתמש בתחביר הבא:

```
<!-- זו תגית הערה -->
```

דוגמאות לשימוש בהערות:

1. **הסבר על חלק מסוים בקוד:**

```
<!-- התפריט הראשי של האתר -->
```

HTML + CSS

block elements - and inline elements:

ההבדל בין div לspan

```
<div style="background-color: lightgray; padding: 15px; font-size: 18px">
  Welcome to our website! Here is some
  <span style="color: blue; font-weight: bold">important information</span> that
  stands out.
</div>

<div style="background-color: lightblue; padding: 10px">
  This is a <strong>div</strong> element. It takes up the entire width and is
  commonly used to create sections in a page.
</div>

<p>
  This is a paragraph with a <span style="color: red">span</span> element
  inside. The <strong>span</strong> only highlights a part of the text and does
  not create a new line.
</p>
```

CSS

הערות
סלקטורים בסיסיים:
Element selector
Class selector
Id Selector

```
/* comment */

h1{
    color: red;
}

.site-header{
    background-color: #333;
    color: #fff;
    padding: 1em;
}

#main-content{
    background-color: #f4f4f4;
    padding: 1em;
}


/* both subtitle and title will be blue */
.title, .subtitle{
    color: blue;
}
```

CSS Combinators

descendant selector (space)
child selector (>)
adjacent sibling selector (+)
general sibling selector (~)

בן, נכד, נין, חימש

```
div p {
  background-color: yellow;
}
```

```
<div>
  <main>
    <p>hello</p>
  </main>
  <p>hello</p>
</div>
```

פסקה בתוך הדיב – כילד ישיר (לא סתם צאצא)

```
div > p {
  background-color: yellow;
}
```

```
<div>
  <p>hello</p>
</div>
```

CSS Combinators

```
<div class="container">
  <section>
    <p id="good">
      Hello, Vite + React!
    </p>
  </section>
</div>
```

.container > section > p#good{}

.container p{}

JS & React

var, const, let, function, arrow function, implicit return, return
DOM = document object model

Nan, Infinity,

אופרטורים בסיסיים:
+, -, *, /, %, +=, -=, ++, --, ===, ==, !=, !==

מושגים

document, document.body

משפטי תנאי, לולאות, משפט תנאי מקוצר (טרנרי)
פונקציות שמקבלות מספר לא ידוע של ארגומנטים

Implicit return:
const add = (a,b)=>a+b

JS & React

Dom methods:
appendChild, getElementById, getElementsByName, getElementsByTagName, querySelector, querySelectorAll, createElement, addEventListener
setAttribute, style,

Element:
innerHTML
Input: input.value;

JS Classes:
Date, Promise, Math,

JS methods:
setTimeout, setInterval, parseInt, parseFloat

JS Promises: (then, catch, finally)

JS Error Handling: try/catch/finally

יצירת תוכנית חדשה של ראקט עם
Create react app

ויצירת תוכנית חדשה של ראקט עם
vite

JS & React

JSX, Component, Hooks, Context API, React Router dom, LocalStorage, VirtualDOM, axios,

העברת מידע מאב לבן : props

מיפוי של רשימות כולל
KEY

useParams
useState להכיר היטב
useEffect להכיר היטב

Conditional rendering
&&

```
{
  3 > 4 ? <div>Loading…</div> : <div>7</div>;
}
```

דוגמא לפונקציה שמקבלת מספר לא ידוע של ארגומנטים

```
function sum(...args) {
  let total = 0;
  for (const arg of args) {
    total += arg;
  }
  return total;
}

console.log(sum(1, 2, 3));
```

```
n = 8;
var x = 1;
var x = 2;
console.log(x); // 2
```

```
2
```

```
undefined
```

```
console.log(n)
```

```
8
```

Implicit return:
const add = (a,b)=>a+b

עוד דוגמאות

```
document.getElementById("runbtn")
.addEventListener("click",
(e)=>{console.log("Clicked")});
```

```
alert(document.getElementById("fname").value)
```

חשוב

```
useEffect(() => {
  // runs after every render
});
useEffect(() => {
  // runs ONCE on mount
}, []);
```

```
function MyComponent() {
  const [counter, setCounter] = useState(0);

  useEffect(() => {
    // runs after every render
    // only if the value of counter changes
    console.log("Counter changed");
  }, [counter]);

  return (
    <div>
      <button onClick={() => setCounter(counter + 1)}>Increment</button>
    </div>
  );
}
```

פונקציה שרצה כשהקומפוננטה מוסרת מהDOM

```
useEffect(() => {
  const handleScroll = (e) => {
    console.log("Scrolled");
  };
  document.addEventListener("scroll", handleScroll);
  // runs after first render
  return () => {
    // runs before the component is removed from the DOM
    document.removeEventListener("scroll",
handleScroll);
  };
}, []);
```

מחזור חיים של קומפוננטה:

```
Mounted
Each Render
UnMount
```

מניעה של רינדור-ללא-הפסקה:

לא נקרא לsetCount(3)
בכל פעם שהקומפוננטה רצה

אלא – נשתמש בuseEffect שירוץ רק פעם אחת.

C# including WPF

מתודות – כולל ערכי ב"מ,
params,
מערכים, מחלקות

מילות גישה:
private, public, protected, readonly

virtual & override

sealed class

מוסכמות שמות למחלקות, מתודות ומשתנים
רב-צורתיות polymorphism

ממשקים ומחלקות אבסטרקטיות

משתנים סטטיים, מתודה סטטית, מחלקה סטטית

Getters and setters
async/await

פעולות LINQ כגון:
Select, Where, Include, OrderBy, Filter, Find

טיפול בשגיאות: try/catch/finally

שגיאת זמן ריצה
שגיאת קומפילציה

חזרו על התגיות המוכרות WPF
Observable Collection
XAML

value Types vs reference types

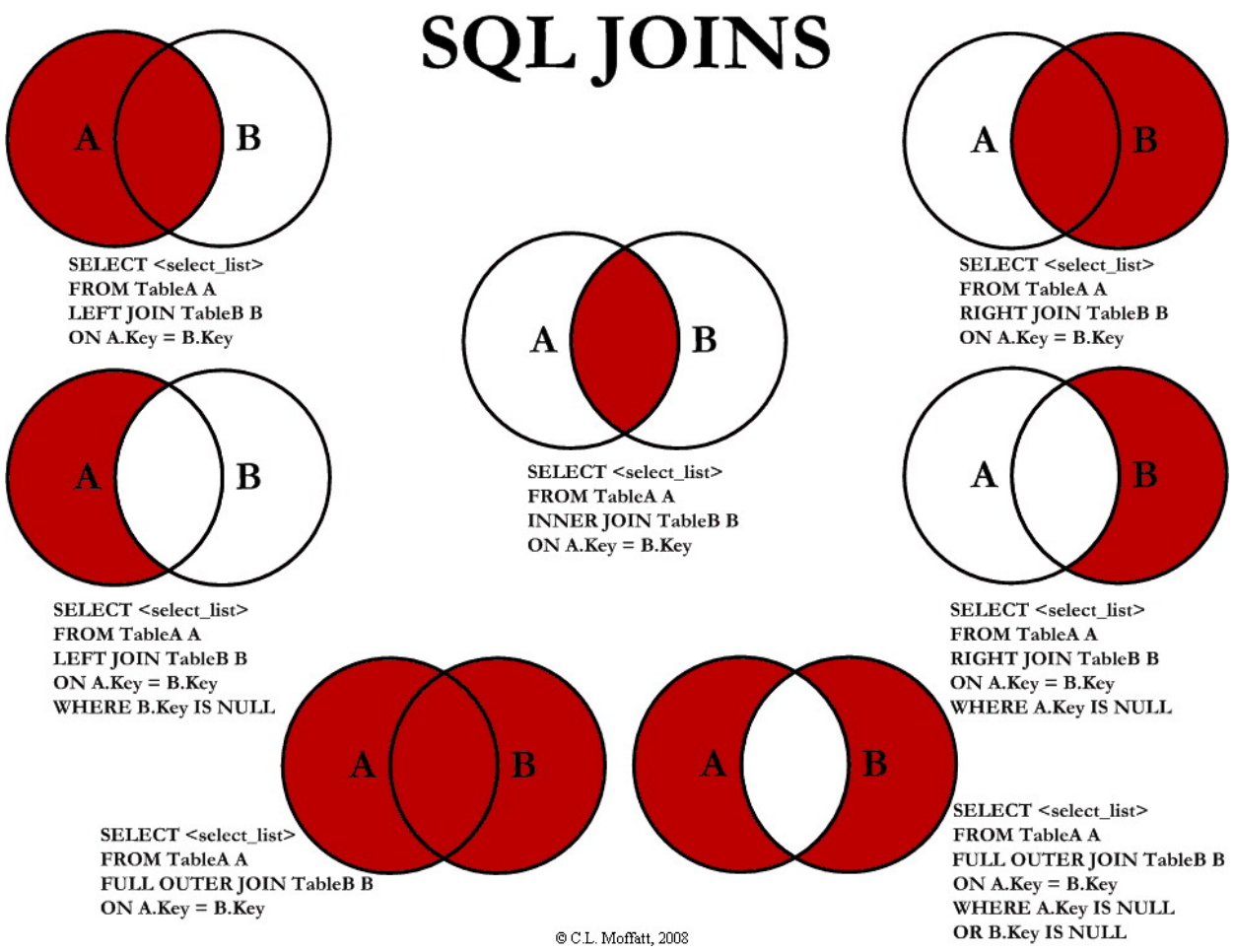
בנאים והעמסת בנאים – המילה this

עבודה עם Threads (תהליכונים)
כולל נעילה של קטע קוד קריטי

משתנים אופציונליים, Nullable
תבנית Singleton
Boxing, Unboxing

JOIN:

סוגי JOIN



SQL:

פעולות
CRUD

לדעת סינטקס מדויק:

INSERT
SELECT
UPDATE
DELETE
ORDER BY
WHERE

SQL:
Group By

SQL:
פונקציות אגרגציה:

SUM, COUNT, AVG, MAX, MIN

Primary Key, Foreign key

סוגי יחסי גומלין:

One to many, many to many, one to one

זיהוי של יחסי גומלין

סוגי בקשות וסטטוסים:

GET
PUT
PATCH
DELETE
POST

200 - Success
201 – Created
400 – Bad request
401 – Unauthorized – לקוח לא מזוהה
403 – Forbidden - לקוח לא מורשה
404 – Not found
405 – Method not allowed
500 – Internal server error

עקרונות מנחים בכתיבת Rest API

מבנה סטנדרטי

מקל על המפתחים שמשתמשים ב API

פורמט אחיד

פורמט הנתונים הוא: JSON/XML

```
[HttpPost]
```

Http methods Status codes

מודולציה וחלוקה לשכבות ביצירת ה API

STATELESS –

בקשה מה הAPI לא צריכה לגרום לשמירת נתונים בזכרון.
מותר לשמור את המידע רק בדטה-בייס.

כל מידע שהוא – אפילו הכי קטן שיש –(מידע על SESSION)

עקרונות של Rest API

Stateless

מבנה אחיד

פורמט נתונים אחיד ונוח - JSON/XML

Status codes

בקשות HTTP

חלוקה למודולים ושכבות

מנגנוני אימות:

Middleware

מאפשר לנו להריץ קוד לפני ואחרי כל בקשה נכנסת

UseAuthentication()
אחראי על אימות המשתמש במערכת מוודא זהות לפני גישה למשאב

UseAuthorization()
אחראי על תפקיד/הרשאות – רק אם למשתמש המאומת יש הרשאות

[Authorize]

[Authorize(roles="admin")]

מנגנוני אימות:

מדיניות CORS (Cross-Origin Resource Sharing) היא מנגנון אבטחה בדפדפנים שמטרתו למנוע גישה למשאבים משרתים שונים על ידי הגבלת בקשות מהמקור (דומיין) הנוכחי בלבד. CORS מוודא שבקשות שנשלחות מדומיין אחד לא יוכלו לגשת לנתונים או לבצע פעולות בשרת בדומיין אחר, אלא אם השרת מאפשר זאת באופן מפורש.

מדיניות זאת מונעת מלקוחות לבצע בקשות אסינכרוניות לדומיין שלא מאפשר זאת.

מנגנוני אימות:

CORS policy:
Cross origin resource sharing

מנגנון אבטחה בדפדפנים – שמגביל לקוח לא לגשת לשרת שלא אפשר את הדומיין של הלקוח

רק אם השרת מחזיר

CORS HEADER
עם הכתובת של הצד-לקוח

הלקוח יוכל לתקשר איתו

מושגים:

DI

DTO
מודל לבקשת לקוח
מודל לתגובה ללקוח
מודל ששמור בדטה-בייס

Service (DBContext, Repository, RoleManager)

appsettings.json תפקידו של הקובץ

מה זה דטה-בייס לא רלציוני
NOSQL

מושגים:

Controller/Service/Model

ASP.NET -

MVC = Model View Controller

Models – מחלקות שכתבנו
View – ממשק משתמש
Controller – מחבר בין הממשק למודלים

מושגים:

What is ASP.NET?

ASP.NET is an open source web framework, created by Microsoft, for building modern web apps and services with .NET.

ASP.NET is cross platform and runs on Windows, Linux, macOS, and Docker.