Topic 1: Intro to `sessionStorage`

نظرة عامة على آلية `sessionStorage` لتخزين البيانات في المتصفح طول مدة الجلسة فقط.

Subtopic 1 — `setItem()`

```
// Save a user name
sessionStorage.setItem('username', 'Ibrahim');
```

`setItem(key, value)` تضيف أو تحدّث زوج مفتاح/قيمة في التخزين.

Subtopic 2 — `getItem()`

```
const username = sessionStorage.getItem('username'); // "Ibrahim"
```

`getItem(key)` ترجع القيمة المرتبطة بالمفتاح أو `null` إن لم يوجد.

Subtopic 3 — `clear()`

```
sessionStorage.clear();   // يمسح كل البيانات
```

Comparison: `sessionStorage` vs `localStorage`

Feature	`sessionStorage`	`localStorage`
Lifetime	Until tab/window closes	Persist across tabs & restarts
Scope	Per-tab	Shared across same-origin tabs
Capacity	~5 MB	~5-10 MB
Use-case	Temp auth data, wizard steps	Preferences, cached API data

---

Topic 2: Basic Auth Flow (`auth/login.js`)

```
/form.onsubmit = e => {
    // 1 Stop the browser from re-loading the page
    e.preventDefault();

    // 2 Grab values the user typed in the inputs
    const email = emailInput.value.trim();     // from <input id="emailInput">
    const password = passwordInput.value;      // from <input id="passwordInput">
```

```
    // 3 Look for a matching account in our dummy data array
    const user = accounts.find(
        acc => acc.email === email && acc.password === password
    );

    if (user) {
        /* 4 Store session data so we can read it on other pages
               -  key:   descriptive string
               -  value: *must* be a string → objects are JSON-stringified
*/
        sessionStorage.setItem('IsLoggedIn', 'true');
        sessionStorage.setItem('UserName', user.username);
        sessionStorage.setItem('UserImg',  user.img);

        // 5 Navigate to the protected page
        window.location.href = 'index.html';
    } else {
        // 6 Invalid creds → show message & reset pwd field
        alert('Invalid email or password. Please try again.');
        passwordInput.value = '';
    }
};
```

شرح عربي مختصر لنفس القسم

شرح خطوة بخطوة لمعالج الإرسال `onsubmit`

الخطوة	الكود	الشرح
1	`e.preventDefault();`	يوقف الإرسال الافتراضي للنموذج حتى نتحكم بالسلوك عبر الجافاسكريبت.
2	`email` / `password`	قراءة القيم الحالية من حقول الإدخال في النموذج (`#emailInput` و`#passwordInput`).
3	`accounts.find(...)`	البحث داخل مصفوفة الحسابات التجريبية `accounts` عن مستخدم يطابق البريد وكلمة المرور.
4	`sessionStorage.setItem(key, value)`	حفظ بيانات الجلسة في المتصفح لعلامة التبويب الحالية فقط. يجب أن تكون القيم نصوصًا؛ إن كانت كائنات نستخدم `JSON.stringify(...)`.
5	`window.location.href`	إعادة توجيه المستخدم بعد نجاح الدخول إلى الصفحة المطلوبة (`index.html`).
6	`alert(...)` + إعادة التعيين	عند الفشل: إظهار رسالة خطأ وتفريغ خانة كلمة المرور.

`setItem()` و`getItem()`

الدالة	المعاملات	القيمة المرجعة	ملاحظات

الدالة	المعاملات	القيمة المرجعة	ملاحظات
`setItem(key, value)`	• `key`: نص • `value`: نص	لا شيء	عند تخزين كائن/مصفوفة حوّلها إلى نص بـ `JSON.stringify`.
`getItem(key)`	• `key`: نص	النص المخزن أو `null`	استخدم `JSON.parse` إذا كانت القيمة المخزنة JSON لاسترجاع الكائن.

ماذا يمكن تخزينه؟

- نصوص (افتراضيًا).
- أرقام/قيم منطقية: خزّنها كنصوص مثل `'42'`, `'true'`.
- كائنات/مصفوفات: استخدم `JSON.stringify` عند الحفظ و`JSON.parse` عند الاسترجاع.
- بيانات مثل: صور Base64، رموز JWT، إلخ، اسم الثيم.
- غير مناسب لتخزين: الدوال، المراجع الدائرية، أو ملفات كبيرة جدًا (عادةً الحد ~5MB حسب المتصفح).

Line-by-Line Explanation of the `onsubmit` Handler

Step	Code	What It Does
1	`e.preventDefault();`	Stops the default form submission so we can handle it with JS.
2	`email` / `password` vars	Reads current values from the form's `<input>` elements.
3	`accounts.find(...)`	Searches our in-memory dummy list (`accounts`) for a user whose email & password both match.
4	`sessionStorage.setItem(key, value)`	Persists login info for this tab only. • All **keys** and **values** are strings. • Objects are stored using `JSON.stringify(obj)` and later retrieved with `JSON.parse(...)`.
5	`window.location.href`	Redirects the logged-in user to the dashboard (`index.html`).
6	`alert(...)` + reset	Shows an error and clears the password field if credentials don't match.

`setItem()` vs `getItem()`

Method	Parameters	Returns	Notes
`setItem(key, value)`	• `key`: string • `value`: string	`void`	If you pass an object/array, use `JSON.stringify` first.
`getItem(key)`	• `key`: string	The stored string **or** `null`	Use `JSON.parse` if the stored value was JSON.

What Can Be Stored?

- **Strings by default.**
- **Numbers / booleans** – convert to strings (`'42'`, `'true'`).
- **Objects & arrays** – `JSON.stringify()` on save, `JSON.parse()` on read.
- **Base64 images, JWTs, theme names**, etc.
- **NOT allowed**: functions, circular references, or blobs > ~5 MB (browser quota).

تتحقق من وجود `isLoggedIn()` عند تسجيل الدخول نحفظ كائن المستخدم في التخزين الجلسي ثم نعيد التوجيه. الدالة المفتاح. `logout()` يمسح التخزين ويعيدك للصفحة الرئيسية.

Topic 3: Header Component (`components/Header.js`)

Below are ready-to-paste sections for your doc. Paste them in `Software/Front-end/Sessions/Session 3/Doc/Frontend-session3.md` immediately AFTER the Topic 3 header code block (after line 198 in your current file).

Header Creation — Element-by-Element

- **navbarEl**: Top-level `<nav>`/`<header>` element. Holds the whole header bar.
- **navbarContainerEl**: Inner container that aligns content and applies Bootstrap classes (spacing, background, layout).
- **homeLinkEl**: Brand/home link. Clicking it returns to the home page.
- **profileWrapperEl**: Right-side container for user info and actions.
- **greetingTextEl**: Text node that shows “Hello, {UserName}”.
- **avatarImgEl**: User avatar image. Styled as a circle with fixed width/height.
- **logoutBtnEl**: Button that logs the user out (clears storage, redirects to login).

Auth Check and Logout Logic

```
// Read current user info
const currentUsername = localStorage.getItem("UserName");
const currentUserImgSrc = localStorage.getItem("UserImg");

// If not logged in, redirect to login page
if (!currentUsername) {
  window.location.href = "login.html";
} else {
  // Show greeting and avatar
  greetingTextEl.textContent = 'Hello,' + currentUsername;
  avatarImgEl.src = currentUserImgSrc;
}

// Logout: clear stored data and go to login
logoutBtnEl.addEventListener("click", () => {
  localStorage.clear();
```

```
  window.location.href = "login.html";
});
```

Line-by-Line Explanation (English)

Line	Code	Explanation
1	`localStorage.getItem("UserName")`	Reads the stored username (string) from storage.
2	`localStorage.getItem("UserImg")`	Reads the stored avatar URL/path.
5	`if (!currentUsername) { ... }`	If no username is found, user is not logged in → redirect.
6	`window.location.href = "login.html";`	Navigate to the login page.
8	`greetingTextEl.textContent = 'Hello,' + currentUsername;`	Show a friendly greeting with the username.
9	`avatarImgEl.src = currentUserImgSrc;`	Attach the user's avatar source to the `<img>`.
13	`logoutBtnEl.addEventListener("click", ...)`	When "Logout" is clicked…
15	`localStorage.clear();`	Remove all stored keys (session footprint).
16	`window.location.href = "login.html";`	Send the user back to the login page.

Arabic Explanation (مختصر)

- **إنشاء العناصر**:
  - `navbarEl`: عنصر الرأس الرئيسي.
  - `navbarContainerEl`: حاوية داخلية للتنسيق والمحاذاة.
  - `homeLinkEl`: رابط الصفحة الرئيسية/العلامة التجارية.
  - `profileWrapperEl`: حاوية يمين تعرض معلومات المستخدم.
  - `greetingTextEl`: نص يظهر التحية والاسم.
  - `avatarImgEl`: صورة المستخدم بشكل دائري.
  - `logoutBtnEl`: زر لتسجيل الخروج.
- **التحقق من تسجيل الدخول**:
  - `localStorage.getItem("UserName")`: يجلب اسم المستخدم المخزن.
  - إذا لم يوجد اسم → تحويل إلى صفحة `login.html`.
  - إذا وُجد → عرض التحية وتعيين صورة المستخدم.
- **تسجيل الخروج**:
  - يتم تنفيذ `Logout` عند الضغط على زر:
    - `localStorage.clear()` لمسح البيانات.

- `window.location.href = "login.html"` للعودة لصفحة تسجيل الدخول.

المكوّن يستورد دوال المصادقة، ويعرض اسم المستخدم أو رابط تسجيل الدخول، ويضيف زرّ الخروج الذي يستدعي `logout()`.

Key Concepts Covered

1. `sessionStorage` methods: `setItem`, `getItem`, `clear`.
2. Stateless vs stateful navigation between pages.
3. Simple auth guard using session data.
4. Building a reusable header component.
5. Differences between `sessionStorage` and `localStorage`.

Practical Tasks

Task A — Build a Register Page (like Login)

- Create `register.html` with a form that includes:
  - Email (required)
  - Password (required)
  - Username (NEW, required)
  - Register button
- On submit:
  - Create a user object: `{ username, email, password, img? }`
  - Store essentials in storage (consistent with your demo: `localStorage`):
    - `IsLoggedIn: 'true'`
    - `UserName: <username>`
    - `UserImg: <optional placeholder URL>`
  - Redirect to `index.html`.
- Use the same header component (`renderHeader()`) on the page you redirect to.

مهمة: أنشئ صفحة تسجيل (مثل صفحة تسجيل الدخول) مع حقل اسم المستخدم الجديد. عند الضغط على زر التسجيل: خزّن بيانات المستخدم (اسم، بريد، كلمة مرور) في التخزين واعد التوجيه للصفحة الرئيسية. استخدم مكوّن الهيدر في الصفحة التالية.

Task B — Reuse Header Component on Another Page

- Pick a second page (e.g., `dashboard.html`) and:
  - Import and call `renderHeader()`
  - Read `UserName` and `UserImg` from storage and show them in the header
  - If not logged in, redirect to `login.html` (same guard as the demo)

مهمة: استخدم مكوّن الهيدر في صفحة ثانية (مثل لوحة التحكم)، واقرأ اسم وصورة المستخدم من التخزين. إذا لم يكن المستخدم مسجلاً، حوّل إلى صفحة تسجيل الدخول.

Acceptance Criteria (Checklist)

- [ ] `register.html` exists with Email, Password, Username fields.
- [ ] On register, storage is updated and user is redirected to `index.html`.
- [ ] `renderHeader()` is used on the redirected page and another page.
- [ ] Header shows “Hello, {UserName}” and avatar; Logout works (clears storage + redirects).

المعايير: صفحة التسجيل تعمل، يتم حفظ البيانات، يتم استخدام الهيدر في أكثر من صفحة، يظهر اسم المستخدم وصورته، وزر الخروج يعمل.

Hints

- Input elements can be built in JS (like your login), or written in HTML with IDs and selected via `document.getElementById(...)`.
- Store objects via `JSON.stringify` and read back via `JSON.parse`.
- Keep storage usage consistent with your demo (you used `localStorage` in login and header). You can later switch to `sessionStorage`.

نصائح: يمكنك إنشاء العناصر بالجافاسكريبت أو كتابتها في HTML. استخدم `JSON.stringify/parse` للكائنات. التزم بنفس التخزين الذي استخدمته في المثال.

شرح تفصيلي للـ Task

- الهدف
  - إنشاء صفحة تسجيل جديدة `register.html` مشابهة لصفحة الدخول، مع حقل إضافي لاسم المستخدم.
  - عند الضغط على زر “Register” يتم حفظ بيانات أساسية في التخزين، ثم التوجيه إلى الصفحة الرئيسية `index.html`.
  - إعادة استخدام مكوّن الهيدر `renderHeader()` في الصفحة التي تنتقل إليها بعد التسجيل (وأي صفحة أخرى).
- المتطلبات الأساسية
  - حقول إدخال: اسم المستخدم، البريد الإلكتروني، كلمة المرور.
  - زر “Register”.
  - تخزين بسيط باستخدام نفس المفاتيح المستعملة في الديمو:
    - `IsLoggedIn: 'true'`
    - `UserName: <username>`
    - `UserImg: <رابط صورة افتراضي>`
  - عند نجاح التسجيل: توجيه إلى `index.html` واستخدام الهيدر لعرض “Hello, {UserName}” والصورة.
- خطوات التنفيذ
  1. إنشاء ملف `register.html` وإضافة نموذج يحتوي على:
     - `input` (إلزامي) لاسم المستخدم).
     - `input` (إلزامي) للبريد الإلكتروني).

    - `input` (إلزامي لكلمة المرور).
    - زر `Register`.
2. ربط الصفحة بسكربت (مثل `register.js`) لمعالجة الإرسال:
    - منع الإرسال الافتراضي للنموذج.
    - قراءة القيم من الحقول.
    - التحقق السريع من أن الحقول غير فارغة.
    - حفظ القيم الأساسية في التخزين بنفس أسلوب الديمو:
        - `localStorage.setItem('IsLoggedIn', 'true')`
        - `localStorage.setItem('UserName', username)`
        - `localStorage.setItem('UserImg', 'path/to/default-avatar.png')`
    - التوجيه إلى `index.html`.
3. في `index.html` (وأي صفحة ثانية مثل `dashboard.html`):
    - استدعاء `renderHeader()` من components/Header.js.
    - التأكد أن الهيدر يقرأ `UserName` و`UserImg` من التخزين ويعرضهما.
    - تحويل إلى `login.html` إن لم يكن المستخدم مسجّلاً الدخول (لا يوجد `UserName`) →

- التحقق من الإنجاز
    - بعد التسجيل؟ هل تظهر تحية "Hello, {UserName}"
    - عند نجاح التسجيل؟ هل يتم التوجيه تلقائياً إلى `index.html`
    - هل يعمل زر الخروج في الهيدر ويمسح التخزين ويعيد إلى `login.html`؟
    - هل يعمل الهيدر بنفس الشكل في صفحة أخرى (مثل `dashboard.html`)؟

- تلميحات واضحة
    - اجعل قيم المدخلات "إلزامية" بإضافة الخاصية `required` في عناصر `input`.
    - استخدم `document.getElementById(...)` أو بناء العناصر ديناميكياً كما فعلت في login.js—اختَر الأسهل لك.
    - اعتمد نفس آلية التخزين التي استعملتها في الديمو (`localStorage`) لضمان اتساق السلوك مع الهيدر الحالي.
    - استخدم صورة افتراضية للمستخدم إن لم يكن لديك رفع صور، مثل:
        - `/assets/img/default-avatar.png` أو رابط صورة علني.
    - في Header.js تأكد من:
        - القراءة بـ `localStorage.getItem('UserName')` و`localStorage.getItem('UserImg')`.
        - التحقق: إذا لم يوجد اسم مستخدم → `window.location.href = 'login.html'`.
        - زر "Logout" يمسح التخزين بـ `localStorage.clear()` ويعيد التوجيه.
    - اختبر السيناريوهات:
        - تسجيل بيانات صحيحة → التوجيه للهوم وعرض الاسم والصورة.
        - تحديث الصفحة بعد التسجيل → يبقى مسجلاً (حتى تمسح التخزين).
        - تسجيل الخروج → العودة لصفحة الدخول.
        - محاولة دخول صفحة محمية بدون تسجيل → إعادة التوجيه للدخول.

- اقتراحات اختيارية (تحسينات سريعة)
    - التحقق من صيغة البريد وكلمة المرور (طول أدنى).
    - تعطيل زر "Register" أثناء المعالجة لتفادي النقر المزدوج.
    - إضافة تنبيه نجاح بسيط قبل التوجيه.
    - إبراز الصفحة الحالية في الهيدر أو إضافة قائمة صغيرة تحت الصورة.