Cookies vs Session Storage vs Local Storage

إيه هما الـ Cookies، Session Storage، و Local Storage؟

لو بتشتغل في تطوير الويب، أكيد سمعت عن الطرق دي لتخزين البيانات على جانب العميل (Client-Side). دول ثلاث آليات أساسية بتساعدنا نحفظ بيانات زي تفضيلات المستخدمين أو جلسات الدخول. كل واحدة ليها خصائص مختلفة، وفهم الفرق بينهم هيساعدك تبني تطبيقات أفضل وأأمن. في المقالة دي، هنقارن بينهم ونشوف إزاي كل واحدة تناسب سيناريوهات معينة.

1. Cookies: القديمة والموثوقة

الـ Cookies هي قطع صغيرة من البيانات بتتخزن في المتصفح. موجودة من سنين طويلة وبتُستخدم لحاجات زي حفظ معلومات المستخدم أو تفضيلاته.

- **الحجم:** محدود جدًا، حوالي 4KB لكل كوكي. يعني مناسبة للبيانات الصغيرة زي Session IDs أو توكنز التصديق Authentication Tokens.
- **المدة:** ممكن تحدد تاريخ انتهاء صلاحيتها. فيه Session Cookies بتختفي لما تقفل المتصفح، وفيه أخرى بتبقى لفترة أطول.
- **الوصول:** متاحة من جانب العميل والسيرفر، يعني بتروح وتيجي مع كل طلب HTTP. ده يخليها مثالية لنقل البيانات بين العميل والسيرفر.
- **الأمان:** عرضة لهجمات زي XSS (Cross-Site Scripting) و CSRF (Cross-Site Request Forgery) لو مش محصنة كويس. لازم تستخدم خيارات زي HttpOnly و Secure عشان تحميها.

مثال: لو عايز تحفظ تفضيلات المستخدم زي اللغة، استخدم كوكيز عشان السيرفر يقدر يقراها ويعمل تعديلات.

2. Session Storage: للجلسات المؤقتة

الـ Session Storage هي ميزة جديدة نسبيًا في المتصفحات، مصممة لحفظ البيانات طول مدة الجلسة الواحدة فقط.

- **المدة:** البيانات بتبقى موجودة بس طول ما الصفحة مفتوحة. لما تقفل التاب أو تخرج من الصفحة، كل حاجة بتتمسح.
- **الحجم:** أكبر من الكوكيز، عادة 5-10 MB لكل دومين. يعني مناسبة لبيانات مؤقتة أكبر شوية زي بيانات النماذج أو إعدادات الجلسة.
- **الوصول:** متاحة بس داخل الصفحة أو التاب نفسها، مش مشتركة بين تابات أو نوافذ مختلفة.
- **الأمان:** أأمن من الكوكيز لأنها مش بتروح مع كل طلب HTTP تلقائيًا، فبتقلل خطر CSRF.

مثال: لو عندك نموذج طويل، استخدم Session Storage عشان تحفظ البيانات مؤقتًا لو المستخدم عمل ريفريش للصفحة.

3. Local Storage: للتخزين الدائم

الـ Local Storage هي طريقة تانية لحفظ البيانات على جهاز المستخدم بشكل دائم.

- **المدة:** البيانات بتبقى موجودة حتى بعد ما تقفل المتصفح، وبتستمر إلى الأبد لحد ما تحذفها يدويًا أو التطبيق يمسحها.
- **الحجم:** كبير، حوالي 5-10 MB لكل دومين. مناسبة لبيانات أكبر زي إعدادات المستخدم أو محتوى مخزن (Cached Content).
- **الوصول:** متاحة عبر كل التابات والنوافذ في نفس المتصفح، يعني سهلة لمشاركة البيانات داخل الموقع نفسه.

🔹 **الأمان:** عامة أأمن، بس عرضة لـ XSS. متخزنش بيانات حساسة هنا بدون تشفير.

مثال: لو عايز تحفظ إعدادات المستخدم زي الثيم الداكن، استخدم Local Storage عشان تبقى موجودة في كل زيارة.

4. إزاي تقرر تستخدم إيه؟

اختيار الطريقة يعتمد على احتياجاتك:

- **Cookies**: للبيانات الصغيرة اللي لازم تنتقل بين العميل والسيرفر، زي توكنز الجلسات أو تفضيلات المستخدم (User Preferences).
- **Session Storage**: للبيانات المؤقتة اللي مش عايزها تبقى بعد الجلسة، زي بيانات نموذج أو إعدادات مؤقتة.
- **Local Storage**: للبيانات الدائمة زي إعدادات المستخدم أو كاش للمحتوى.

أمثلة كود للـ Cookies

مثال كود للـ Cookies في ASP.NET Core

في Controller، بنستخدم Response.Cookies لحفظ، و Request.Cookies للقراءة.

```
using Microsoft.AspNetCore.Mvc;

[ApiController]
[Route("[controller]")]
public class CookiesController : ControllerBase
{
    // دالة لحفظ كوكي
    [HttpPost("set")]
    public IActionResult SetCookie(string name, string value, int?
days = null)
```

```
    {
        var cookieOptions = new CookieOptions
        {
            Expires = days.HasValue ?
DateTime.Now.AddDays(days.Value) : null,
            HttpOnly = true, // للأمان
            Secure = true // يتطلب HTTPS
        };
        Response.Cookies.Append(name, value, cookieOptions);
        return Ok($"تم حفظ الكوكي: {name} = {value}");
    }

    // دالة لقراءة كوكي
    [HttpGet("get/{name}")]
    public IActionResult GetCookie(string name)
    {
        if (Request.Cookies.TryGetValue(name, out var value))
        {
            return Ok($"قيمة الكوكي: {value}");
        }
        return NotFound(";("الكوكي غير موجود
{

// دالة لحذف كوكي
    [HttpDelete("delete/{name}")]
    public IActionResult DeleteCookie(string name)
    {
        Response.Cookies.Delete(name);
        return Ok($"تم حذف الكوكي: {name}");
    }
}
```

مثال الاستخدام: استخدم Postman أو Swagger للـ POST ./Cookies/set?name=username&value=Ahmed&days=7، ثم GET ./Cookies/get/username. الكوكيز هتروح مع الطلبات للسيرفر.

مثال كود للـ Cookies في Javascript

الـ Cookies بتحتاج شوية كود أكتر عشان هي مش زي الـ Storage APIs. هنستخدم دالة بسيطة لحفظ وقراءة الكوكيز.

```
// دالة لحفظ كوكي
function setCookie(name, value, days) {
    let expires = "";
    if (days) {
        const date = new Date();
        date.setTime(date.getTime() + (days * 24 * 60 * 60 * 1000));
        expires = "; expires=" + date.toUTCString();
    }
    document.cookie = name + "=" + (value || "") + expires + ";
path=/";
}

// دالة لقراءة كوكي
function getCookie(name) {
    const nameEQ = name + "=";
    const ca = document.cookie.split(';');
    for (let i = 0; i < ca.length; i++) {
        let c = ca[i];
        while (c.charAt(0) === ' ') c = c.substring(1, c.length);
        if (c.indexOf(nameEQ) === 0) return
c.substring(nameEQ.length, c.length);
    }
    return null;
}

// دالة لحذف كوكي
function deleteCookie(name) {
    document.cookie = name + '=; Max-Age=-99999999;';
}

// مثال الاستخدام
setCookie('username', 'Ahmed', 7); // حفظ لـ 7 أيام
console.log(getCookie('username')); // الناتج: Ahmed
```

```
deleteCookie('username'); // حذف
console.log(getCookie('username')); // الناتج: null
```

ده مثال بسيط: نحفظ اسم مستخدم، نقراه، ونمسحه. تذكر إن الكوكيز بتروح مع الطلبات للسيرفر.

مثال كود لل Session Storage

مثال كود لل Session Storage في Blazor

استخدم ProtectedSessionStorage لحفظ البيانات المؤقتة (بتمسح لما تقفل التاب).

أولاً، أضف في Program.cs:

```
builder.Services.AddScoped<ProtectedSessionStorage>();
```

ثم، في Razor Component (مثل Counter.razor):

```
@page "/counter"
@using Microsoft.AspNetCore.Components.ProtectedBrowserStorage
@inject ProtectedSessionStorage ProtectedSessionStore
@inject IJSRuntime JSRuntime

<h1>Session Storage Example</h1>

<button @onclick="SaveToSession">حفظ في Session</button>
<button @onclick="LoadFromSession">قراءة من Session</button>
<button @onclick="DeleteFromSession">حذف من Session</button>

<p>@message</p>

@code {
    private string message = "";
```

```
    private async Task SaveToSession()
    {
        await ProtectedSessionStore.SetAsync("theme", "dark");
        message = "تم حفظ 'dark' في Session Storage";
    }

    private async Task LoadFromSession()
    {
        var result = await
ProtectedSessionStore.GetAsync<string>("theme");
        message = $"قيمة Session: {result.Success ? result.Value : "غير
موجود"}";
    }

    private async Task DeleteFromSession()
    {
        await ProtectedSessionStore.DeleteAsync("theme");
        message = "تم حذف البيانات من Session Storage";
    }
}
```

مثال الاستخدام: شغل الصفحة، اضغط الأزرار. البيانات هتختفي لما تقفل التب.

مثال كود للـ Session Storage في Javascript

الـ Session Storage أسهل، عشان هي API مباشرة في الـ window.

```
// حفظ بيانات
sessionStorage.setItem('theme', 'dark');

// قراءة بيانات
const theme = sessionStorage.getItem('theme');
console.log(theme); // الناتج: dark

// حذف بيانات
sessionStorage.removeItem('theme');
console.log(sessionStorage.getItem('theme')); // الناتج: null
```

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

هنا نحفظ ثيم مؤقت، نقراه، ونمسحه. البيانات هتختفي لما تقفل التب.

مثال كود لل Local Storage

مثال كود لل Local Storage في Blazor

شبه Session، بس بـ ProtectedLocalStorage (بتبقى دائمة).

أضف في Program.cs:

```
builder.Services.AddScoped<ProtectedLocalStorage>();
```

في Razor Component:

```
@page "/local"
@using Microsoft.AspNetCore.Components.ProtectedBrowserStorage
@inject ProtectedLocalStorage ProtectedLocalStore
@inject IJSRuntime JSRuntime

<h1>Local Storage Example</h1>

<button @onclick="SaveToLocal">حفظ في Local</button>
<button @onclick="LoadFromLocal">قراءة من Local</button>
<button @onclick="DeleteFromLocal">حذف من Local</button>

<p>@message</p>

@code {
    private string message = "";

    private async Task SaveToLocal()
    {
        await ProtectedLocalStore.SetAsync("language", "arabic");
```

```
        message = "تم حفظ 'arabic' في Local Storage";
    }

    private async Task LoadFromLocal()
    {
        var result = await
ProtectedLocalStore.GetAsync<string>("language");
        message = $"قيمة Local: {result.Success ? result.Value : "غير
موجود"}";
    }

    private async Task DeleteFromLocal()
    {
        await ProtectedLocalStore.DeleteAsync("language");
        message = "تم حذف البيانات من Local Storage";
    }
}
```

مثال الاستخدام: شغل الصفحة، البيانات هتبقى موجودة حتى بعد إعادة فتح المتصفح.

مثال كود للـ Local Storage في Javascript

شبه الـ Session Storage، بس البيانات بتبقى دائمة.

```
// حفظ بيانات
localStorage.setItem('language', 'arabic');

// قراءة بيانات
const language = localStorage.getItem('language');
console.log(language); // الناتج: arabic

// حذف بيانات
localStorage.removeItem('language');
console.log(localStorage.getItem('language')); // الناتج: null

// حذف كل البيانات
```

```
localStorage.clear();
```

مثال: نحفظ لغة، نقراها، ونمسحها. البيانات هتبقى موجودة حتى بعد إعادة فتح المتصفح.

نصايح مهمة للتعامل معهم

✔️ **انتبه للأمان:** دايمًا استخدم HTTPS مع الكوكيز، وتجنب حفظ بيانات حساسة في Local أو Session Storage بدون تشفير.

✔️ **احترم خصوصية المستخدم:** عرف المستخدمين عن استخدام الكوكيز (زي في GDPR)، ومتعتمدش عليهم لو المتصفح بيمنعهم.

✔️ **اختبر الحجم:** لو البيانات كبيرة، استخدم Local أو Session Storage بدل الكوكيز عشان متوصلش للحد.

✔️ **استخدم API سهلة:** في JavaScript، استخدم document.cookie للكوكيز، sessionStorage.setItem() للسيشن، و localStorage.setItem() للوكال.

✔️ جرب الكود في console المتصفح عشان تشوف النتايج فورًا.

✔️ لو عايز تستخدم في مشروع حقيقي، أضف error handling عشان حالات زي عدم دعم المتصفح.