Server-side Validation Vs Client-side Validation

إيه هو الـ Validation؟

لو بتشتغل في تطوير الويب أو التطبيقات، أكيد سمعت عن الـ Validation، اللي هي عملية التحقق من صحة البيانات اللي بيبعتها المستخدم. زي لما حد يسجل في موقع، لازم تتأكد إن الإيميل صحيح، الباسورد قوي، والحقول مش فاضية. الـ Validation ده بيتم في مكانين رئيسيين: جانب العميل (Client-side) أو جانب الخادم (Server-side). خلينا نفهم الفرق بينهم بطريقة بسيطة، وإيه اللي يناسب مشروعك.

الموضوع ده مهم جدًا عشان يحمي التطبيق من الأخطاء، الهجمات، ويحسن تجربة المستخدم.

1. إيه هو Client-side Validation؟

الـ Client-side Validation هو التحقق اللي بيحصل على جهاز المستخدم نفسه، يعني داخل المتصفح أو التطبيق المحلي. ده بيتم باستخدام لغات زي JavaScript، أو حتى خصائص HTML5 زي required أو pattern.

🔹 إزاي بيشتغل؟

لما المستخدم يدخل بيانات في Form، التحقق بيحصل فوريًا قبل ما البيانات تبعث للخادم.

مثال: لو حد دخل إيميل غلط، هيظهر رسالة خطأ على طول من غير ما يضغط Submit.

🔹 المزايا

سريع جدًا: مش بيحتاج تبعت Request للـ Server، فالتجربة سلسة ومش بتتأخر.

بيحسن UX (User Experience): المستخدم يشوف الأخطاء فوريًا، زي "الباسورد قصير جدًا".

يقلل الحمل على الـ Server: لأنه مش بيبعت Request مع كل محاولة Submission.

🔹 العيوب

غير آمن: أي حد يعرف برمجة يقدر يتجاوزه بسهولة، زي باستخدام Developer Tools في المتصفح.

مش موثوق 100%: لو المستخدم عطل JavaScript، التحقق ده هيبطل يشتغل.

محدود: مش بيقدر يتحقق من حاجات زي إذا الإيميل ده موجود في الداتابيز أصلاً.

مثال عملي في HTML/JS:

```
<form>
  <input type="email" id="email" required
pattern="[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,}$">
  <button type="submit">Submit</button>
</form>

<script>

document.getElementById('email').addEventListener('input',
function() {
```

```
    if (!this.checkValidity()) {
      alert('إيميل غلط!');
    }
  });

</script>
```

هنا التحقق بيحصل فوريًا في المتصفح.

2. إيه هو Server-side Validation؟

الـ Server-side Validation هو التحقق اللي بيحصل على الخادم بعد ما البيانات تتبعت.

🔹 كيف بيشتغل؟

البيانات تتبعت للـ Server عبر POST أو API، والخادم يتحقق منها قبل ما يخزنها في الداتابيز أو يعمل أي عملية.

مثال: الخادم يشوف إذا الـ Username ده موجود قبل كده، أو يتحقق من CAPTCHA لمنع الهجمات.

🔹 المزايا

آمن جدًا: مش بيتجاوز بسهولة، لأنه على الـ Server اللي تحت سيطرتك.

موثوق: بيشتغل حتى لو العميل عطل JavaScript أو حاول يخدع النظام.

قوي: يقدر يتحقق من حاجات معقدة زي الوصول للداتابيز أو خدمات خارجية.

🔹 العيوب

أبطأ: بيحتاج Request، فلو في خطأ، المستخدم هيستنى الرد من الـ Server.

يزود الحمل على الـ Server: كل طلب بيحتاج معالجة، خاصة لو في هجمات زي DDoS.

تجربة أقل سلاسة: الأخطاء مش هتظهر فوريًا، لازم يرجع رد من الخادم.

مثال عملي في ASP.NET Core:

```
using Microsoft.AspNetCore.Mvc;

[ApiController]
[Route("api/[controller]")]
public class RegisterController : ControllerBase
{
    [HttpPost]
    public IActionResult Post([FromBody] RegisterModel model)
    {
        if (string.IsNullOrEmpty(model.Email) ||
!model.Email.Contains('@'))
        {
            return BadRequest(new { error = ";({ "إيميل غلط!
{
// هنا تحقق من الداتابيز إذا الإيميل موجود
        return Ok(new { success = "تم التسجيل!" });
    }
}

public class RegisterModel
```

```
{
    public string Email { get; set; }
}
```

هنا التحقق بيحصل على الخادم، ولو خطأ، يرجع رد JSON.

3. الفرق بين Server-side و Client-side Validation

الاثنين مهمين، بس مش نفس الشيء. خلينا نقارن بالنسبة لـ :

- المكان

- **Client-side Validation:** على الـ Browser / الـ Client
- **Server-side Validation:** على الـ Server

- السرعة

- **Client-side Validation:** سريع جدًا
- **Server-side Validation:** أبطأ بسبب الـ Request

- الأمان

- **Client-side Validation:** ضعيف (سهل التجاوز)
- **Server-side Validation:** قوي (موثوق)

- التعامل مع الأخطاء

- **Client-side Validation:** فوري
- **Server-side Validation:** بعد الـ Request

- الاعتماد

- **Client-side Validation:** على JavaScript/HTML
- **Server-side Validation:** على لغة الـ Back-end

- الاستخدام

- **Client-side Validation:** لتحسين UX
- **Server-side Validation:** للأمان والدقة

من الآخر: الـ Client-side زي "البواب الأول" اللي يمنع الأخطاء البسيطة، بس الـ Server-side هو "المدير" اللي يتأكد من كل حاجة.

🔹 امتى تستخدمهم؟

استخدم Client-side للأشياء البسيطة زي التحقق من طول الحقل أو تنسيق الإيميل.

استخدم Server-side دائمًا للأمان، زي منع SQL Injection أو تحقق من الهوية.

الأفضل: استخدم الاثنين مع بعض! Client-side للسرعة، و Server-side للأمان. وكمان بيبقي في خط دفاع تالت وهو الـ Database Validation من خلال الـ Constraints علي الـ Tables والـ Relationships بينهم. عشان يبقي الـ Application بتاعك Secure.

طيب الناس بتتلخبط في إيه؟

افتراض إن Client-side كفاية: ده غلط، لأنه مش آمن ضد الهجمات زي XSS أو CSRF.

عدم التحقق مرة تانية على الـ Server: لو اعتمدت بس على Client-side، حد يقدر يبعث بيانات غلط مباشرة عبر API.

نسيان الـ Edge Cases: زي التحقق من NULL أو قيم غير متوقعة.

نصايح مهمة للتعامل مع الـ Validation

✔️ **دائمًا زد طبقة أمان:** استخدم مكتبات زي Validator.js للـ Client-side، و Express-Validator للـ Server-side في Node.js. أو Attribute Validation في Asp.Net.

✔️ **راجع الأخطاء بوضوح:** أرسل رسائل خطأ مفهومة للمستخدم، عشان يفهم بوضوح ايه الخطأ اللي عمله.

✔️ **اختبر جيدًا:** جرب سيناريوهات زي انك توقف JavaScript أو إرسال بيانات ضارة.

✔️ **اعتمد على Best Practices:** دور علي الطريقة الصح عشان تطبق كل نوع Validation.