LECTURE 4

Semantic HTML and HTML Forms

Organizing a page so it means something — to humans, browsers, and search engines — and collecting user input.

CSC336 Web Technologies

OVERVIEW

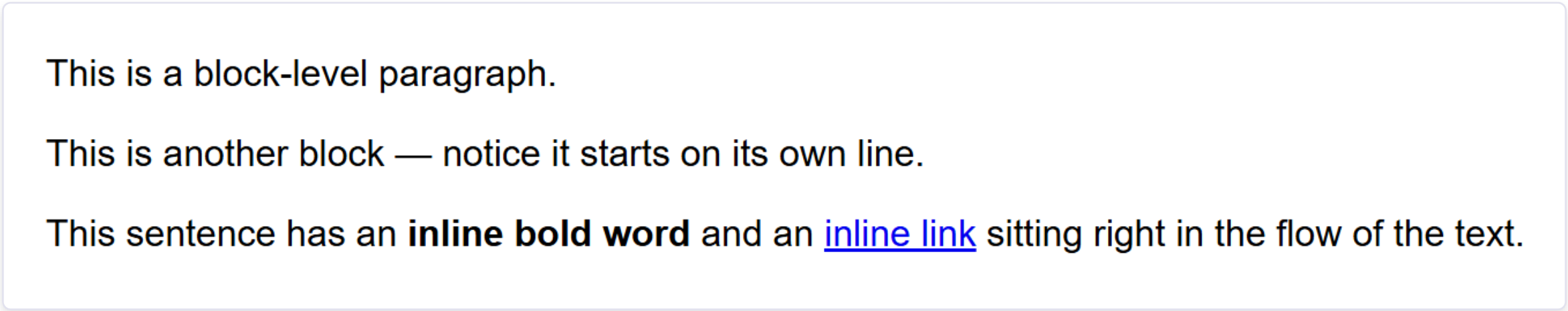
In This Lecture

1. Block-level vs. inline elements, and the generic /
2. HTML5 semantic elements, and why they matter for accessibility and SEO
3. Build forms: <form>, action/method — including a real Google search GET form
4. <input> types, <select>, <textarea>, <button>
5. Upload files with multipart/form-data
6. Client-side validation with required, pattern, min, max

Block-Level vs. Inline (Recap)

Block elements each claim their own line; inline elements flow within the surrounding text.

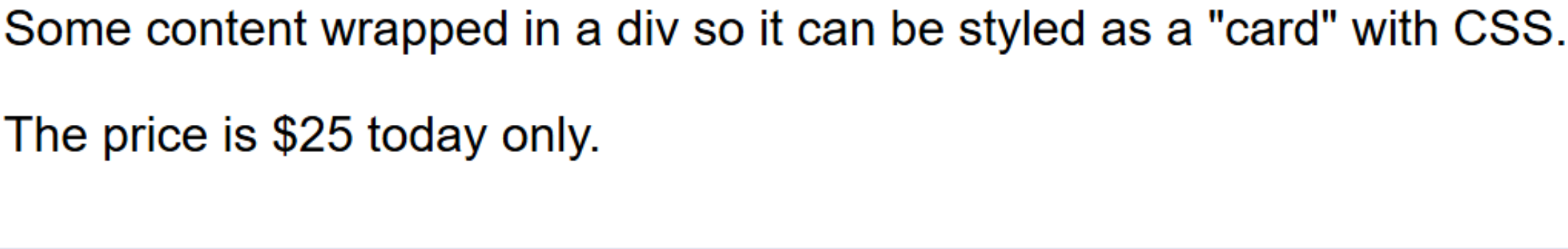
```
<p>This is a block-level paragraph.</p>
<p>This is another block — starts on its own line.</p>

<p>This sentence has an <strong>inline bold word</strong> and an
<a href="#">inline link</a> in the flow of the text.</p>
```



The Generic and

* is a generic BLOCK container; is a generic INLINE container — neither means anything on its own.*

```
<div class="card">
    <p>Some content wrapped in a div so it can be styled as a "card".</p>
</div>

<p>The price is <span class="highlight-price">$25</span> today only.</p>
```



Meaningless Divs vs. Semantic Tags

WITHOUT SEMANTICS (OLD STYLE)

-
-
-
-

WITH SEMANTICS (HTML5)

- <header>
- <nav>
- <main>
- <footer>

The Main Semantic Elements

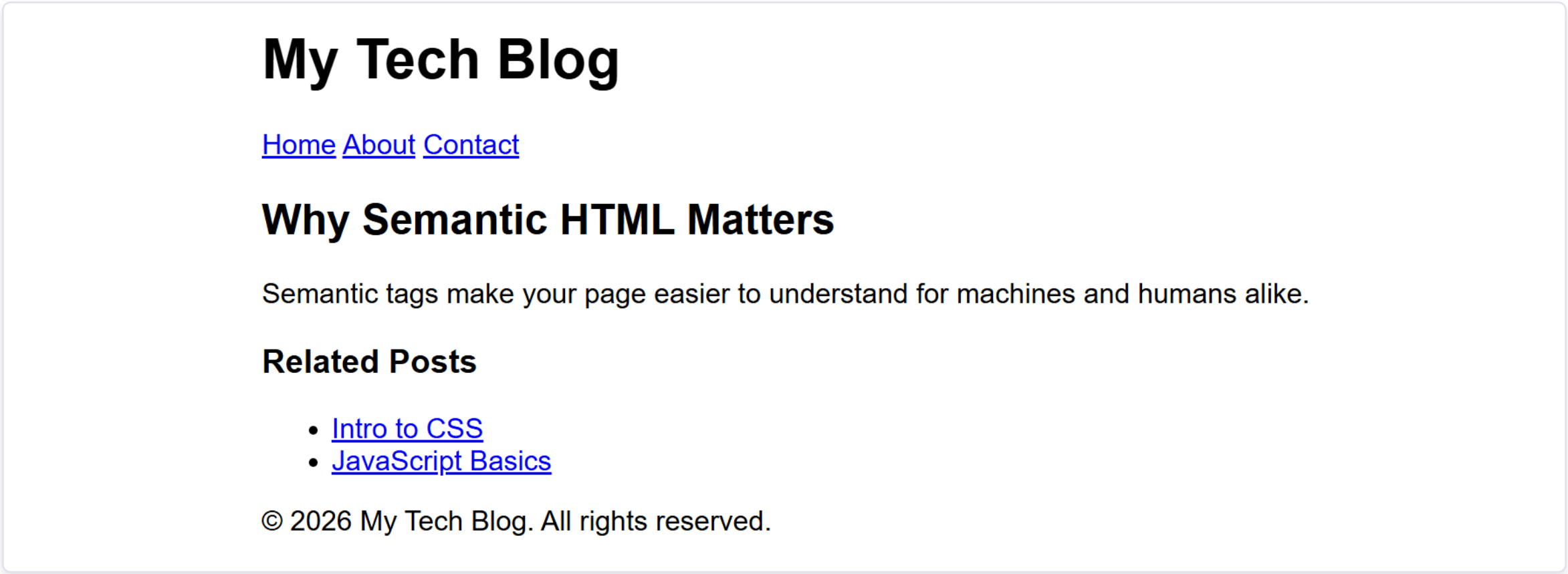
Element	Represents
<header>	Introductory content — logo, title, nav
<nav>	A block of navigation links
<main>	The primary, unique content (only one per page)
<section> / <article>	A themed grouping / self-contained standalone content
<aside>	Content tangentially related (a sidebar)
<footer>	Closing content — copyright, contact links

A Full Page Skeleton

```
<header>
    <h1>My Tech Blog</h1>
    <nav><a href="/">Home</a> <a href="/about.html">About</a></nav>
</header>
<main>
    <article><h2>Why Semantic HTML Matters</h2><p>Semantic tags help everyone.</p></article>
    <aside><h3>Related Posts</h3></aside>
</main>
<footer><p>&copy; 2026 My Tech Blog.</p></footer>
```

A Full Page Skeleton, Rendered

Semantic elements have no built-in visual style — but every section genuinely exists in the markup.



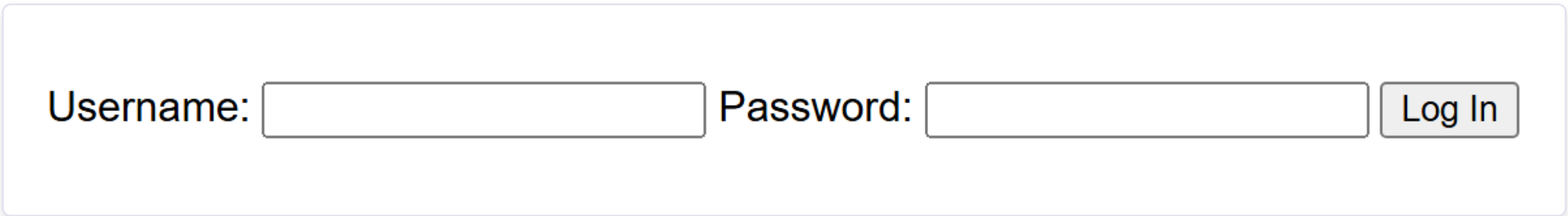
WHY IT MATTERS

Accessibility and SEO

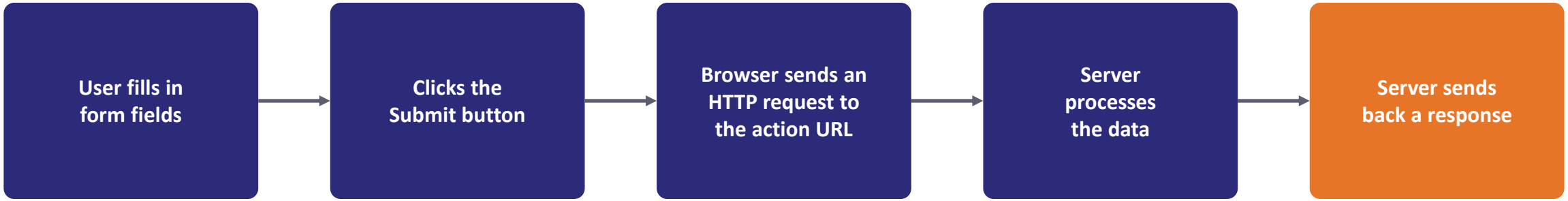
- Accessibility: screen readers use semantic tags to jump to navigation or skip to main content — a div-only page gives them nothing to work with
- SEO: search engines weight content inside <article>/<main>/headings more than anonymous s
- Rule of thumb: reach for a semantic element first; fall back to / only when nothing else fits

A Basic Form

```
<form action="/submit-login" method="POST">
    <label for="username">Username:</label>
    <input type="text" id="username" name="username">

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

    <button type="submit">Log In</button>
</form>
```



What Happens on Submit



The action URL and the method (GET or POST) decide where the request goes and how the form's data is packaged inside it.

The action and method Attributes

Attribute	Meaning
action	The URL the form's data is sent to on submit
method: GET	Appends data to the URL as a query string — for searches, not sensitive data
method: POST	Sends data in the request body — for anything that creates/changes data, or is sensitive

SECURITY

Never Use GET for Passwords

WARNING

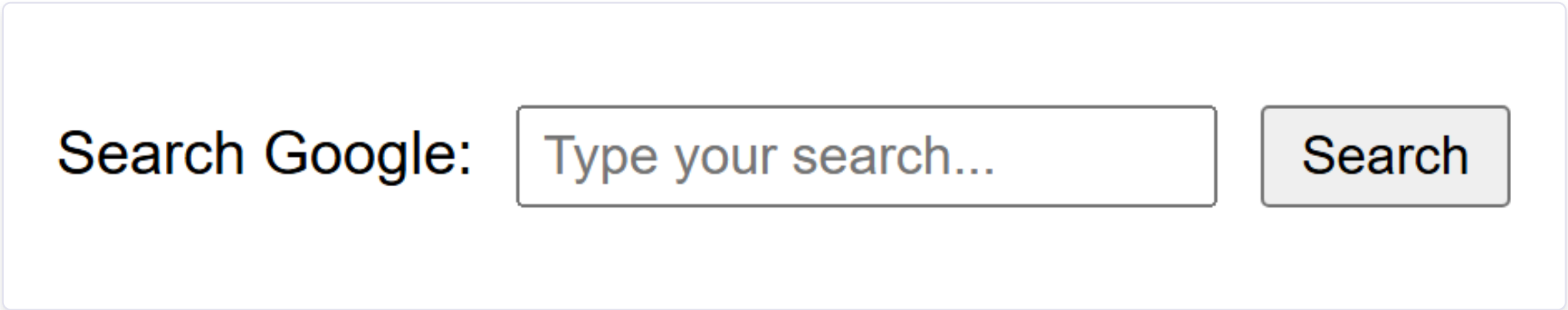
GET puts form values directly into the URL, where they can end up saved in browser history, server logs, and shared links. Always use POST for passwords and other sensitive information.

A REAL EXAMPLE

Searching Google with a GET Form

name="q" is the exact query-parameter name Google's search endpoint expects — submitting this form genuinely searches Google.

```
<form action="https://www.google.com/search" method="GET">
    <label for="q">Search Google:</label>
    <input type="text" id="q" name="q" placeholder="Type your search...">
    <button type="submit">Search</button>
</form>
```



Common <input> Types

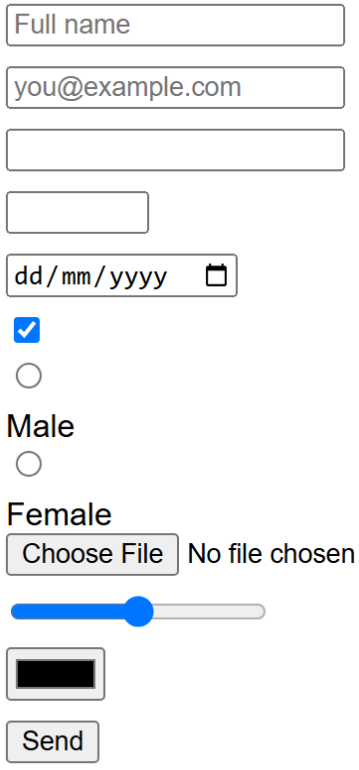
Type	Purpose
text / email / password	A line of text, checked for an email shape, or hidden as dots
number / date	Numeric input with arrows / a date picker
checkbox / radio	An on/off box / a mutually-exclusive choice (same name)
file	Lets the user pick a file to upload
range / color	A slider / a color picker
submit / hidden	Submits the form / not shown, but sent along with the data

Every <input> Type

```
<input type="text" name="fullname" placeholder="Full name">
<input type="email" name="email" placeholder="you@example.com">
<input type="password" name="password">
<input type="number" name="age" min="1" max="120">
<input type="date" name="birthday">
<input type="checkbox" name="subscribe" checked>
<input type="radio" name="gender" value="male"> Male
<input type="radio" name="gender" value="female"> Female
<input type="file" name="resume">
<input type="range" name="volume" min="0" max="100">
<input type="color" name="favcolor">
<input type="submit" value="Send">
```

Every Input Type, Rendered

Each type is a genuinely different native control — this is the browser doing most of the work for you.



multipart/form-data

- A file's raw bytes can't fit in a URL — file uploads require method="POST"
- enctype="multipart/form-data" switches the request body format so text fields and raw file bytes can travel together
- accept is a filter hint for the file picker, not a security check — the server must always re-validate the uploaded file

What multipart/form-data Looks Like

```
POST /upload-resume HTTP/1.1
Content-Type: multipart/form-data; boundary=----Bnd123

------Bnd123
Content-Disposition: form-data; name="resume"; filename="cv.pdf"

%PDF-1.4 ...(raw binary bytes)...
------Bnd123--
```

Node.js/Express (with multer) parses this format for you automatically — you'll never write boundary-splitting code by hand.

<select>, <textarea>, and <button>

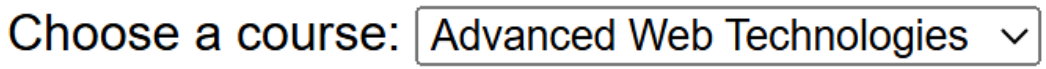
1. <select> + <option> creates a dropdown; the selected attribute pre-picks one
2. <textarea> is a resizable, multi-line text box — its default text goes between the tags, not in a value attribute
3. <button> is more flexible than <input type="submit"> because it can contain other HTML

<select> and <textarea> Markup

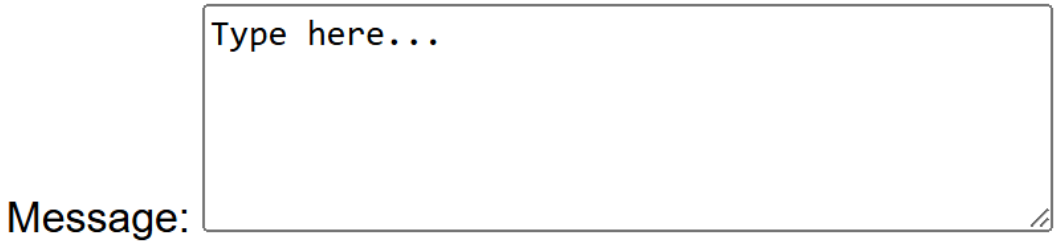
```
<select id="course" name="course">
    <option value="csc336">Web Technologies</option>
    <option value="csc337" selected>Advanced Web Technologies</option>
</select>

<textarea id="message" name="message" rows="5" cols="40">Type here...</textarea>
```

Dropdown and Textarea, Rendered

SELECT



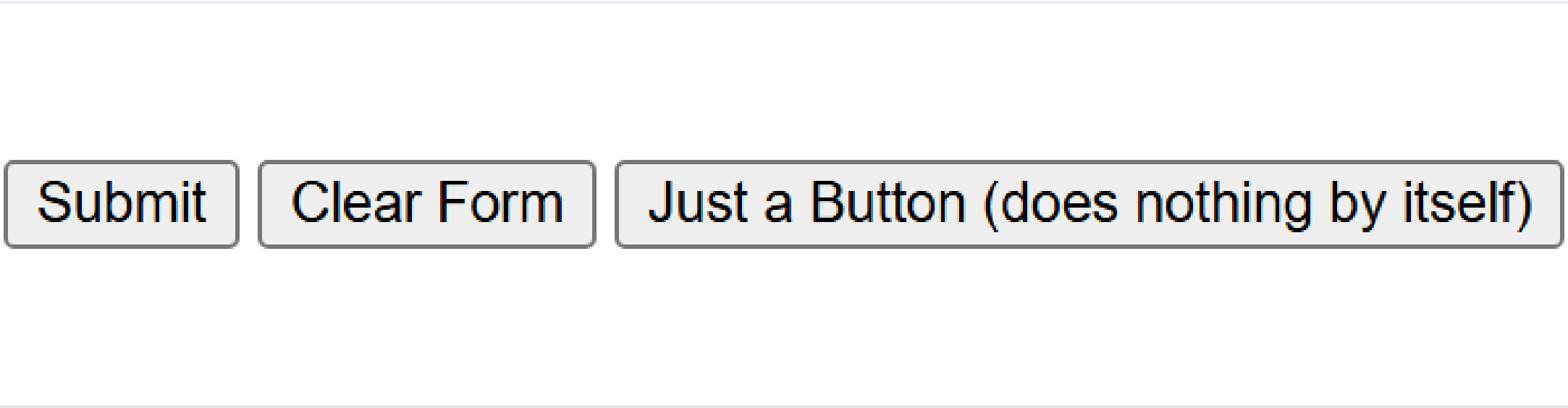
TEXTAREA



Three Button Types

submit submits the form, reset clears it, button does nothing on its own (wired up with JavaScript later).

```
<button type="submit">Submit</button>
<button type="reset">Clear Form</button>
<button type="button">Just a Button (does nothing by itself)</button>
```



HTML5 Client-Side Validation

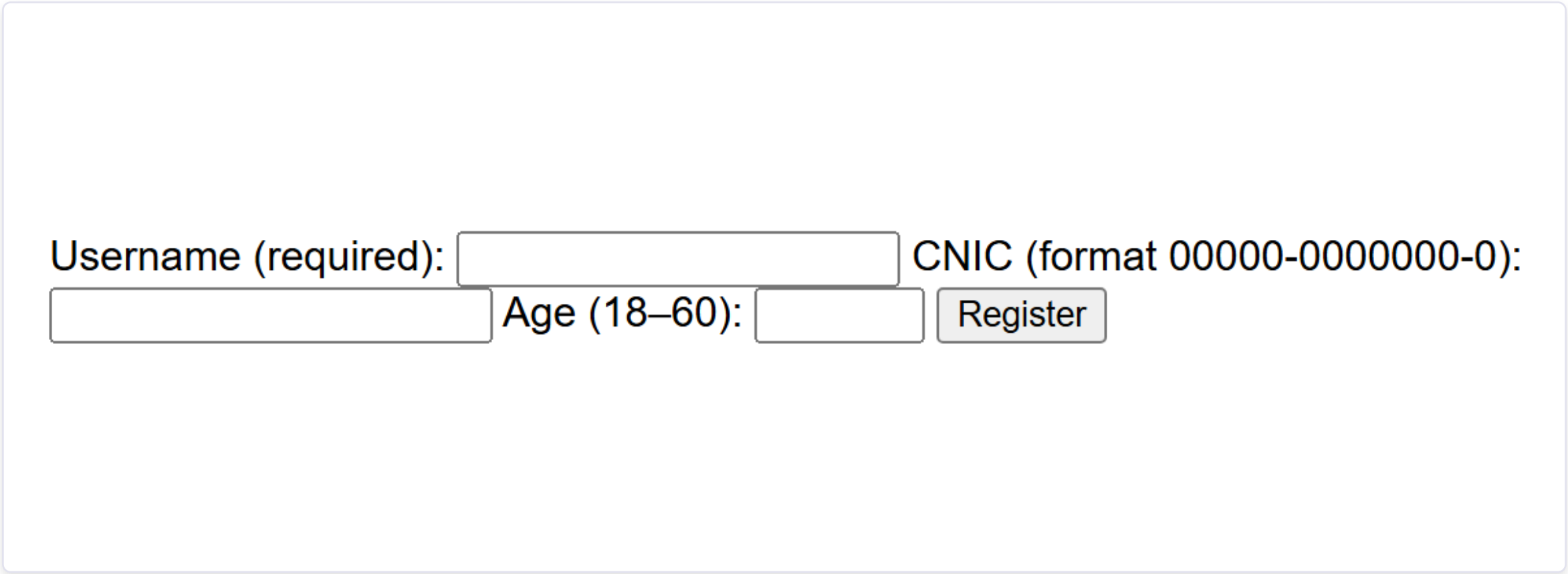
Attribute	Effect
required	The field cannot be left empty
pattern	A regular expression the value must match (with title for a hint)
min / max / step	Lowest/highest acceptable value and allowed increment
maxlength / minlength	Character-count limits on a text field

A Validated Registration Form

```
<form action="/register" method="POST">
    <label for="uname">Username (required):</label>
    <input type="text" id="uname" name="uname" required>

    <label for="cnic">CNIC (format 00000-0000000-0):</label>
    <input type="text" id="cnic" name="cnic"
           pattern="\d{5}-\d{7}-\d" title="Format: 00000-0000000-0">

    <label for="age">Age (18-60):</label>
    <input type="number" id="age" name="age" min="18" max="60">

    <button type="submit">Register</button>
</form>
```

A Validated Form, Rendered

The warning bubbles only appear live in a browser when you try to submit invalid or missing data.



Client-Side Validation Is Not Enough

WARNING

A user can disable JavaScript or edit HTML in dev tools. Client-side validation is a convenience for instant feedback — it is not security. Every submission must also be validated again on the server.

KEY TAKEAWAYS

Lecture 4 in Six Points

1. Semantic elements (header, nav, main, article, footer) describe role, improving accessibility and SEO over generic divs.
2. action sets where form data goes; method (GET/POST) sets how — GET builds a query string, exactly how Google's search box works.
3. <input type="..."> covers most controls; <select>, <textarea>, and <button> cover the rest.
4. File uploads need method="POST" and enctype="multipart/form-data" — accept is a hint, not a security check.
5. required/pattern/min/max give instant feedback, but must always be backed by server-side validation.
6. Always pair inputs with <label>, connected via matching for/id attributes.