8. 폼(form)과 사용자 입력

8.1 폼 태그 구조 (`<form>`, `action`, `method`)

HTML 폼은 사용자로부터 **입력 데이터를 수집**하고 서버에 **전송**하기 위한 구조다.
로그인, 회원가입, 검색, 주문 등 거의 모든 웹 애플리케이션에서 필수적인 요소이며,
폼의 기본 구조는 `<form>` 태그와 그 속성인 `action`, `method`로 시작된다.

✅ 기본 구조

```
<form action="/submit" method="post">
  <label for="username">아이디:</label>
  <input type="text" id="username" name="username">

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

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

✅ 각 구성 요소 설명

🔹 `<form>`

- 폼의 **전체 컨테이너**
- 안에 들어가는 `<input>`, `<textarea>`, `<select>`, `<button>` 등이 모두 포함됨

🔹 `action`

- 폼 데이터를 제출할 **서버 주소(URL)**
- 상대 경로 또는 절대 경로 가능

```
<form action="/join">       <!-- 현재 서버의 /join으로 제출 -->
<form action="https://api.example.com/regist"> <!-- 외부 주소 제출 -->
```

🔹 `method`

- 데이터를 전송하는 HTTP **방식(method)**
- `GET` 또는 `POST` 사용 가능

방식	특징
`GET`	URL에 데이터가 붙음 (`?name=abc`) / 북마크 가능 / 보안에 취약 / 캐싱됨
`POST`	본문(body)에 데이터 포함 / 보안에 유리 / 로그인 등 민감한 정보에 사용

✅ method 차이 예시

🟢 GET

```
<form action="/search" method="get">
  <input name="q" type="text" />
  <button type="submit">검색</button>
</form>
```

➡️ 결과 요청: `/search?q=사용자입력값`

🟡 POST

```
<form action="/login" method="post">
  <input name="user" />
  <input name="pw" type="password" />
  <button type="submit">로그인</button>
</form>
```

➡️ 사용자가 입력한 정보는 **HTTP 요청 본문(body)** 에 담겨 전송됨
➡️ URL에 노출되지 않기 때문에 로그인, 결제 등에 적합

✅ 기타 주의사항

속성	설명
`enctype`	폼 데이터 인코딩 방식 (파일 업로드 시 사용)
`novalidate`	브라우저 기본 유효성 검사 비활성화
`target`	응답 받을 창 (`_self`, `_blank`, 등)

예: 파일 업로드 시

```
<form action="/upload" method="post" enctype="multipart/form-data">
```

✅ 폼 제출 버튼

- `<button type="submit">` 또는 `<input type="submit">`
- 버튼 클릭 → `form` 태그의 `action`으로 전송됨

✅ 자바스크립트와의 연동

```
<form onsubmit="return validateForm()">
```

- 자바스크립트로 폼 유효성 검사 후 `false`를 리턴하면 전송 막을 수 있음

✅ 한 줄 요약

> `<form>`은 사용자의 데이터를 수집하고 서버로 전송하기 위한 구조이며, `action`은 전송 위치, `method`는 전송 방식(`GET` 또는 `POST`)을 지정하는 핵심 속성이다.

8.2 텍스트 입력 (`<input type="text">`, `placeholder`)

`<input type="text">`는 HTML에서 가장 기본적인 입력 필드로,
한 줄짜리 텍스트를 입력받을 때 사용된다.
사용자 이름, 이메일, 검색어 등 다양한 입력을 처리하는 기본 수단이다.

✅ 기본 문법

```
<input type="text" name="username">
```

속성	설명
`type="text"`	텍스트 입력 필드 지정
`name`	서버에 전송될 데이터의 키 이름
`value`	초기값 설정
`placeholder`	입력 전 안내 텍스트
`maxlength`	최대 입력 문자 수 제한
`readonly`	읽기 전용 (수정 불가)
`disabled`	입력 비활성화
`required`	필수 입력 필드로 지정 (유효성 검사)
`autocomplete`	자동완성 사용 여부 지정

✅ 예제 1: 가장 단순한 입력 필드

```
<input type="text" name="nickname">
```

✅ 예제 2: `placeholder` 사용

```
<input type="text" name="email" placeholder="이메일을 입력하세요">
```

- 사용자 입력 전에는 흐릿한 안내 문구가 보인다.
- 입력하면 placeholder는 사라진다.

✅ 예제 3: 초기값 지정 (`value`)

```
<input type="text" name="city" value="Seoul">
```

- 사용자가 처음 폼을 열었을 때 자동으로 `"Seoul"`이 입력된 상태가 된다.

✅ 예제 4: 입력 길이 제한 (`maxlength`)

```
<input type="text" name="zip" maxlength="5" placeholder="우편번호 5자리">
```

- 최대 5글자만 입력 가능 (초과 시 자동 차단)

✅ 예제 5: 읽기 전용 & 비활성화

```
<!-- 읽기 전용 -->
<input type="text" value="변경 불가" readonly>

<!-- 완전 비활성화 -->
<input type="text" value="회색 처리됨" disabled>
```

- `readonly`: 값은 보이고 복사 가능하지만 수정 불가
- `disabled`: 회색으로 비활성화되며, **폼 제출 시 전송되지 않음**

✅ 예제 6: 필수 입력 지정 (`required`)

```
<form>
  <input type="text" name="user" required placeholder="아이디를 입력하세요">
  <button type="submit">제출</button>
</form>
```

- 사용자가 입력하지 않으면 제출할 수 없음
- 브라우저가 자동으로 경고 메시지 제공

✅ 예제 7: 자동완성 기능 (`autocomplete`)

```
<!-- 이전에 입력했던 검색어 자동완성 제공 -->
<input type="text" name="search" autocomplete="on">

<!-- 자동완성 방지 -->
<input type="text" name="search" autocomplete="off">
```

✅ 접근성을 위한 `label` 연결

```
<label for="name">이름</label>
<input type="text" id="name" name="name">
```

- `label`과 `input`을 연결해 스크린 리더 등 접근성 개선
- `for="id"`를 통해 연동

✅ 브라우저 예시 동작

속성 조합	효과
`placeholder="이메일 입력"`	안내 문구 표시
`value="홍길동"`	자동 입력값
`maxlength="10"`	10자 제한
`required` + `type="text"`	빈칸 제출 시 브라우저 경고
`readonly` / `disabled`	읽기 전용 or 비활성화

✅ 한 줄 요약

> `<input type="text">`는 사용자로부터 한 줄의 문자열을 입력받는 가장 기본적인 입력 필드이며, `placeholder`, `value`, `maxlength`, `required` 등을 활용하여 UX를 향상시킬 수 있다.

8.3 다양한 입력 필드 (`email`, `password`, `tel`, `url`, `number`)

HTML5에서는 사용자 입력을 보다 **정확하고 의미 있게 표현**하기 위해
`<input>` 태그의 `type` 속성에 다양한 **입력 유형**이 도입되었다.
각 입력 타입은 브라우저나 모바일 키보드에서 **전용 UI를 제공**하며,
기본적인 **입력 유효성 검사(Validation)** 도 함께 지원한다.

✅ 전체 요약

타입	입력 예시	특징 및 유효성 검사 기준
`email`	`user@example.com`	이메일 형식 (`@` 포함) 검사
`password`	`******`	입력값을 가림 (별표로 표시)
`tel`	`010-1234-5678`	전화번호 전용, 형식 검사 X
`url`	`https://example.com`	URL 형식 검사 (`http://` 등)
`number`	`123`, `3.14`, `-5`	숫자만 입력, ↑↓ 스피너 제공

✅ 1. 이메일 입력 (`type="email"`)

```
<input type="email" name="email" placeholder="이메일 입력" required>
```

- `@` 기호와 도메인이 포함된 **올바른 이메일 형식**인지 검사
- 브라우저가 자동으로 유효성 체크

📌 모바일: 이메일 키보드 제공 (`@`, `.` 강조)

✅ 2. 비밀번호 입력 (`type="password"`)

```
<input type="password" name="password" placeholder="비밀번호 입력">
```

- 입력한 문자를 **숨김 처리**(● 또는 • 표시)
- 비밀번호 길이 제한은 `minlength`, `maxlength` 사용 가능

```
<input type="password" minlength="8" maxlength="20">
```

✅ 3. 전화번호 입력 (`type="tel"`)

```
<input type="tel" name="phone" placeholder="010-1234-5678">
```

- 브라우저는 **형식 검사를 하지 않음**
- 하지만 모바일에서는 **전화번호 키패드**가 뜬다
- 유효성 검사는 패턴으로 직접 지정해야 함

```
<input type="tel" pattern="^\d{3}-\d{4}-\d{4}$" required>
```

✅ 4. URL 입력 (`type="url"`)

```
<input type="url" name="homepage" placeholder="https://example.com">
```

- `http://`, `https://` 등의 **URL 형식**이 요구됨
- 잘못된 형식이면 제출 시 브라우저가 차단함

✅ 5. 숫자 입력 (`type="number"`)

```
<input type="number" name="age" min="0" max="120" step="1" required>
```

속성	설명
`min`	최소값 제한
`max`	최대값 제한
`step`	증감 단위 (예: 0.1, 5 등)
`value`	초기값 설정

- 마우스로 ↑↓ **스피너** 제공
- 음수/소수 등 입력 가능 (제한하려면 추가 설정 필요)

✅ 예시: 회원가입용 다양한 필드

```
<form>
  <label>이메일</label>
  <input type="email" name="email" required><br>

  <label>비밀번호</label>
  <input type="password" name="password" required><br>

  <label>전화번호</label>
  <input type="tel" name="phone"><br>

  <label>홈페이지</label>
  <input type="url" name="homepage"><br>

  <label>나이</label>
  <input type="number" name="age" min="0" max="150"><br>

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

✅ 한 줄 요약

> `input`의 다양한 `type` 속성 (`email`, `password`, `tel`, `url`, `number`)을 사용하면 입력 목적에 맞는 **전용 UI와 유효성 검사**를 통해 사용자 경험을 향상시킬 수 있다.

8.4 버튼 (`submit`, `reset`, `button`)

HTML에서 `<button>` 또는 `<input>` 태그를 사용하여
폼 제출, 초기화, 사용자 지정 동작을 수행할 수 있다.
버튼의 종류는 `type` 속성에 따라 다음 3가지로 나뉜다:

타입	설명
`submit`	폼 데이터를 서버로 전송
`reset`	폼의 입력값을 초기 상태로 되돌림
`button`	기본 동작 없음 (JavaScript와 함께 사용)

✅ 1. `type="submit"`: 폼 제출 버튼

```
<form action="/login" method="post">
  <input type="text" name="user">
  <button type="submit">로그인</button>
</form>
```

- 버튼 클릭 시 **form의 `action` 경로**로 데이터를 전송한다.
- 기본값이므로 `type` 생략 시 `submit`으로 처리된다.

```
<button>제출</button> <!-- submit과 동일 -->
```

📌 Enter 키를 눌렀을 때도 자동으로 `submit`이 트리거됨 (input이 하나일 경우)

✅ 2. `type="reset"`: 폼 초기화 버튼

```
<form>
  <input type="text" name="name" value="홍길동">
  <input type="email" name="email" value="test@example.com">
  <button type="reset">초기화</button>
</form>
```

- 클릭하면 폼의 모든 입력값이 **초기 상태(value 값)** 로 되돌아감
- 서버와의 통신은 없음

✅ 3. `type="button"`: 사용자 정의 버튼

```
<button type="button" onclick="alert('클릭됨!')">알림</button>
```

- 아무 동작도 하지 않으며 **JavaScript와 함께 동작** 정의
- 버튼 클릭 시 `submit`되지 않도록 하기 위해 자주 사용됨

✅ 비교: `<button>` vs `<input>`

태그 종류	예시	차이점
`<input type="submit">`	`<input type="submit" value="전송">`	콘텐츠를 `value`로만 설정
`<button type="submit">`	`<button>전송</button>`	안에 HTML 포함 가능 (`<span>`, `<img>` 등)

✅ 스타일과 구성 예시

```
<button type="submit" style="background: green; color: white;">
  ✅ 가입하기
</button>
<button type="reset" style="background: gray;">
  ❌ 초기화
</button>
<button type="button" onclick="window.location.href='/'">
  🔙 홈으로
</button>
```

✅ 주의: `<button>`은 기본값이 `submit`

```
<form>
  <button>의도치 않게 제출됨</button> <!-- 기본 submit -->
</form>
```

➡️ 의도치 않은 제출을 방지하려면 명시적으로 `type="button"` 설정

✅ 한 줄 요약

> `submit`, `reset`, `button`은 각각 **폼 전송**, **폼 초기화**, **사용자 정의 동작**을 담당하는 버튼이며, 올바른 타입 지정이 중요하다.

8.5 라디오 버튼, 체크박스

HTML 폼에서 **선택형 입력 필드**를 만들 때 가장 많이 사용하는 요소는
라디오 버튼과 **체크박스**다.
사용자는 둘 중 하나 또는 여러 개의 항목을 선택할 수 있으며,
이를 통해 **성별, 관심사, 약관 동의** 등 다양한 UI를 구성할 수 있다.

✅ 라디오 버튼 (`type="radio"`)

- **여러 개 중에서 하나만 선택**할 수 있음
- 같은 `name` 속성을 가진 라디오 버튼끼리 **그룹화**됨

```
<p>성별 선택:</p>
<input type="radio" id="male" name="gender" value="M">
<label for="male">남성</label>

<input type="radio" id="female" name="gender" value="F">
<label for="female">여성</label>
```

- `name="gender"`: 하나의 그룹
- 하나를 선택하면 다른 하나는 자동으로 해제됨
- `value`는 서버에 전송될 실제 값

📌 `checked` 속성을 사용하면 **초기 선택값** 설정 가능

```
<input type="radio" name="gender" value="M" checked>
```

✅ 체크박스 (`type="checkbox"`)

- **다수 선택 가능**
- 각 항목은 독립적이며 `name`이 같아도 상관 없음

```
<p>관심 분야 선택:</p>
<input type="checkbox" id="dev" name="interest" value="dev">
<label for="dev">개발</label>

<input type="checkbox" id="design" name="interest" value="design">
<label for="design">디자인</label>

<input type="checkbox" id="marketing" name="interest" value="marketing">
<label for="marketing">마케팅</label>
```

- 여러 개를 동시에 선택할 수 있음
- 서버로 전송될 때 동일한 `name`으로 여러 `value`가 전송됨 (배열처럼 처리)

✅ 기본 선택 상태 (`checked`)

```
<input type="checkbox" name="tos" value="agree" checked> 이용약관 동의
```

- 체크 상태로 초기화하려면 `checked` 속성 사용

✅ 라디오 버튼 vs 체크박스 비교

항목	라디오 버튼 (`radio`)	체크박스 (`checkbox`)
선택 수	단일 선택 (1개만)	복수 선택 가능
그룹 지정	`name` 으로 그룹화	`name` 으로도 그룹 가능 (필수 아님)
UI	원형 버튼	사각형 체크박스
사용 예	성별, 옵션 선택	관심사, 알림 설정

✅ 접근성과 UX를 위한 `label` 연동

```
<input type="radio" id="option1" name="choice" value="1">
<label for="option1">옵션 1</label>
```

- `label` 을 클릭하면 해당 `input` 이 선택됨
- 모바일 사용성 향상 및 스크린 리더 호환

✅ 자바스크립트와 함께 활용

```
<input type="checkbox" id="allCheck" onclick="toggleAll()">
<label for="allCheck">모두 선택</label>

<script>
  function toggleAll() {
    const checkboxes = document.querySelectorAll('input[type="checkbox"]
[name="interest"]');
    const all = document.getElementById('allCheck').checked;
    checkboxes.forEach(cb => cb.checked = all);
  }
</script>
```

✅ 서버 전송 시 구조

```
<input type="checkbox" name="interest" value="dev" checked>
<input type="checkbox" name="interest" value="design">
```

➡️ 서버로 전송: `interest=dev`

➡️ 여러 개 선택 시: `interest=dev&interest=design`

✅ 한 줄 요약

> 라디오 버튼은 단일 선택, 체크박스는 복수 선택을 위한 입력 요소이며, `checked`, `name`, `value`, `label`을 적절히 구성해야 한다.

8.6 드롭다운(`select`, `option`)

드롭다운 목록은 사용자가 **여러 옵션 중 하나를 선택**할 수 있게 해주는
폼 요소로, 공간을 적게 차지하면서도 **선택지를 깔끔하게 구성**할 수 있다.
HTML에서는 `<select>`와 `<option>`, `<optgroup>` 태그로 구성한다.

✅ 기본 구조

```
<select name="country">
  <option value="kr">대한민국</option>
  <option value="us">미국</option>
  <option value="jp">일본</option>
</select>
```

태그	설명
`<select>`	드롭다운 전체 박스
`<option>`	선택 가능한 항목
`value` 속성	서버에 전송될 실제 값

✅ 초기 선택값 지정: `selected`

```
<option value="kr" selected>대한민국</option>
```

- 해당 항목이 **처음부터 선택된 상태**로 표시됨

✅ 사용자 선택 필수: `required`

```
<select name="color" required>
  <option value="">색상을 선택하세요</option>
  <option value="red">빨강</option>
  <option value="green">초록</option>
</select>
```

- 첫 번째 option에 `value=""`을 설정하면 **유효성 검사에 실패하도록 유도** 가능

✅ 드롭다운 그룹화: `<optgroup>`

```
<select name="car">
  <optgroup label="국산차">
    <option value="hyundai">현대</option>
    <option value="kia">기아</option>
  </optgroup>
  <optgroup label="외제차">
    <option value="bmw">BMW</option>
    <option value="audi">Audi</option>
  </optgroup>
</select>
```

- `label` 속성은 그룹 이름으로 표시됨
- 항목들을 **범주별로 시각적으로 구분**할 수 있다

✅ 다중 선택: `multiple`

```
<select name="language" multiple size="4">
  <option value="python">Python</option>
  <option value="java">Java</option>
  <option value="cpp">C++</option>
  <option value="js">JavaScript</option>
</select>
```

- 여러 항목을 **Ctrl(Windows)/Cmd(Mac)** 또는 **Shift 키**와 함께 선택 가능
- `name` 속성에 `[]` 추가해 배열처럼 서버로 전송 가능:

```
<select name="language[]" multiple>
```

✅ `<select>` vs `<input type="radio">`

항목	`<select>`	`radio`
UI 크기	작고 컴팩트	많아지면 공간 차지
선택 수	보통 1개 (`multiple` 가능)	1개 (다중 선택 불가)
사용 예시	국가 선택, 카테고리	성별, 하나의 고정된 선택 항목들

✅ 접근성과 UX

- `<label for="select-id">`로 명확한 설명 제공
- `<option>` 안에 길고 구체적인 설명을 넣으면 **스크린 리더**에서 도움됨

✅ 한 줄 요약

> `<select>`는 옵션 선택을 위한 드롭다운 UI이며, `<option>`으로 항목을 구성하고, `<optgroup>`으로 카테고리를 그룹화할 수 있다. `multiple` 속성으로 다중 선택도 지원한다.

8.7 멀티라인 입력(`textarea`)

`<textarea>`는 사용자로부터 **여러 줄의 텍스트 입력**을 받을 수 있게 해주는 HTML 폼 요소다.
한 줄만 입력 가능한 `<input type="text">`와 달리, 블로그 글쓰기, 문의사항, 주소 등
긴 내용을 입력받는 데 최적화되어 있다.

✅ 기본 구조

```
<label for="message">메시지를 입력하세요:</label><br>
<textarea id="message" name="message"></textarea>
```

- 태그 안의 콘텐츠가 **기본 텍스트**로 표시됨
- 입력된 내용은 서버 전송 시 `name` 속성 값으로 전송됨

✅ 주요 속성 정리

속성	설명
`name`	폼 데이터 전송 시 사용되는 키
`rows`	초기 세로 줄 수 (행)
`cols`	초기 가로 너비 (열)
`placeholder`	힌트 텍스트 표시

속성	설명
`maxlength`	최대 입력 길이 제한
`required`	필수 입력 필드 지정
`readonly`	읽기 전용
`disabled`	비활성화 상태
`wrap`	줄 바꿈 방식 (`soft`, `hard`, `off`)

✅ 예제 1: 기본 사용

```
<textarea name="comment" rows="5" cols="30">
이곳에 의견을 입력해주세요.
</textarea>
```

📌 이처럼 `<textarea>` 안에 미리 텍스트를 넣으면 기본값으로 표시됨

✅ 예제 2: placeholder 사용

```
<textarea name="memo" rows="4" cols="50" placeholder="메모를 입력하세요..."></textarea>
```

📌 placeholder는 `<textarea>` 열었을 때 **입력 힌트**를 제공한다. (기본 텍스트와 다름)

✅ 예제 3: maxlength 제한

```
<textarea name="description" maxlength="300"></textarea>
```

- 사용자가 300자 이상 입력할 수 없음

✅ 예제 4: wrap 속성

```
<textarea name="summary" wrap="soft"></textarea>
```

값	설명
`soft`	줄 바꿈 없이 서버에 원문 그대로 전송
`hard`	사용자가 줄 바꿈한 그대로 `\n` 포함 전송
`off`	줄 바꿈 비활성화 (스크롤만 허용)

✅ `<textarea>` vs `<input type="text">`

항목	`<textarea>`	`<input type="text">`
입력 가능 줄	여러 줄	한 줄
태그 구조	여는/닫는 태그 필요	단일 태그
기본 텍스트	내부 텍스트	`value` 속성 사용
사용 예시	글쓰기, 설명, 주소	이름, 전화번호 등

✅ 접근성과 레이아웃 팁

- `<label for="id">` 와 함께 사용하여 폼 설명을 명확히 제공
- CSS로 `resize: none;` 을 설정하면 크기 조절 비활성화 가능

```
textarea {
  resize: none;
}
```

✅ 한 줄 요약

> `<textarea>`는 여러 줄의 텍스트를 입력받기 위한 폼 요소로, `rows`, `cols`, `placeholder`, `maxlength` 등의 속성을 통해 크기와 동작을 제어할 수 있다.

8.8 `label`과 접근성

`<label>` 태그는 **폼 요소에 대한 설명(라벨)**을 제공하여 사용자가
입력 필드의 용도를 쉽게 이해하도록 돕는다.
또한, 웹 접근성(특히 **스크린 리더 사용자**나 **키보드 사용자**)을 위해 반드시 필요한 요소다.

✅ 기본 사용법

```
<label for="username">아이디</label>
<input type="text" id="username" name="username">
```

- `for="username"` → 연결할 `<input>`의 `id="username"`을 참조
- 사용자가 `label`을 클릭하면 **연결된 input이 자동으로 포커스됨**

✅ `<label>`의 2가지 사용 방식

방식	특징 및 예시
명시적 연결 (explicit)	`for` 속성 사용 → 명확하게 특정 input을 참조
암시적 연결 (implicit)	`label` 내부에 input을 포함

✅ 명시적 연결 (권장)

```
<label for="email">이메일</label>
<input type="email" id="email" name="email">
```

- 접근성 최상 (스크린 리더, 키보드 접근 완벽 대응)

✅ 암시적 연결

```
<label>
  이메일
  <input type="email" name="email">
</label>
```

- `for` 없이 **포함 구조로 연동**
- 간단하지만, 명시적 방식보다 명확성이 떨어짐

✅ 필수 입력 표시 예시 (`<span>` 활용)

```
<label for="password">
  비밀번호 <span style="color:red;">*</span>
</label>
<input type="password" id="password" required>
```

- 시각적 표시뿐 아니라 **접근성을 위해** `aria-required="true"` **등**도 함께 쓰는 것이 좋음

✅ `<label>`과 스크린 리더

- 스크린 리더는 `<label>`과 연결된 `input`의 **의미와 내용을 정확히 전달**해줌
- `placeholder`만 사용하는 경우, 접근성 도구에서는 **안 읽히는 경우가 많음**

```
<!-- 비추: 접근성 떨어짐 -->
<input type="text" placeholder="이메일 입력">

<!-- 추천 -->
<label for="email">이메일</label>
<input type="text" id="email" name="email" placeholder="이메일 입력">
```

✅ 여러 항목에 동일한 label 연결 ❌

```
<!-- 이렇게 하면 label은 오작동함 -->
<label for="a">공통 라벨</label>
<input id="a" type="text">
<input id="a" type="text">
```

- `id`는 고유해야 하며, 하나의 `<label>`은 하나의 input만 참조해야 한다.

✅ `<input>` 없이 `<label>`만 사용 가능할까?

- 가능하지만 의미가 **불명확해지고 접근성 저하**됨
- `<label>`은 **폼 컨트롤에 대한 설명자 역할**을 하므로, 반드시 연결된 대상이 있어야 함

✅ 디자인 요소와 label

- `checkbox`, `radio`는 `label`과 조합 시 **UI/UX를 획기적으로 향상**시킬 수 있음

```
<input type="checkbox" id="tos">
<label for="tos">이용약관에 동의합니다.</label>
```

- `<label>`을 누르면 체크됨 → 모바일에서 특히 중요

✅ 한 줄 요약

> `<label>`은 입력 요소에 의미를 부여하고 접근성을 높이는 핵심 태그로, `for` 속성을 사용해 관련 입력 필드와 명확히 연결하는 것이 좋다.

8.9 필수 입력, 기본값, 최대/최소 (`required`, `value`, `min`, `max`)

HTML 폼에서 입력 필드의 **필수 여부, 기본값, 허용 범위** 등을 지정하는 속성들은
사용자 입력의 유효성을 제어하고, **서버 부하를 줄이며**,
사용자 경험(UX)을 개선하는 데 매우 중요하다.

✅ 1. `required`: 필수 입력 필드

```
<input type="text" name="username" required>
```

- 사용자가 값을 입력하지 않으면 폼 제출 불가
- 브라우저가 자동으로 **경고 메시지**를 표시함
- `<select>`, `<textarea>`, `<input>` 등에 사용 가능

📌 `required`는 Boolean 속성 → `required="required"`처럼 써도 되고, `required`만 써도 됨

✅ 2. `value`: 기본값 설정

```
<input type="text" name="nickname" value="홍길동">
```

- 입력 필드에 **초기값을 자동으로 채움**
- 사용자가 수정하지 않아도 해당 값이 전송됨
- 드롭다운이나 라디오, 체크박스에도 적용 가능

```
<input type="radio" name="gender" value="M" checked> 남성
```

✅ 3. `min`, `max`: 숫자/날짜 범위 제한

이 속성들은 숫자 입력, 날짜 입력 등에서 **허용 범위를 지정**할 수 있다.

🎯 숫자 입력

```
<input type="number" name="age" min="18" max="65">
```

- 18보다 작거나 65보다 큰 값은 입력 불가 (브라우저가 유효성 검사)

🎯 날짜 입력

```
<input type="date" name="birthday" min="1900-01-01" max="2025-12-31">
```

- 해당 날짜 범위 외에는 선택 불가

📌 `type="range"` 슬라이더에도 `min`, `max`, `step` 같이 사용 가능

✅ 4. `step`: 증감 단위 지정

```
<input type="number" name="score" min="0" max="100" step="5">
```

- 0, 5, 10, 15... 처럼 5 단위로만 선택 가능

✅ 예시 통합

```
<form>
  <label for="price">가격 (1000~10000):</label>
  <input type="number" id="price" name="price" min="1000" max="10000" required><br><br>

  <label for="username">이름 (기본값 포함):</label>
  <input type="text" id="username" name="username" value="사용자"><br><br>

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

✅ 브라우저 기본 유효성 검사 메시지

- `required` 입력 안 하면: **"이 필드는 필수입니다."**
- `min` 이하 값: **"값이 너무 작습니다."**
- `max` 초과 값: **"값이 너무 큽니다."**

📌 **JavaScript 없이도 검증 가능**
단, 고급 검증이 필요한 경우에는 JS로 `input.validity` 또는 `setCustomValidity()` 활용

✅ 서버 측 유효성 검사는 필수!

HTML의 `required`, `min`, `max`는 **클라이언트 측 검증**일 뿐이므로
서버에서도 동일한 유효성 검사를 반드시 해야 한다.

✅ 한 줄 요약

> `required`, `value`, `min`, `max` 속성은 입력 필드의 기본값 설정과 유효성 제어를 위한 핵심 속성으로, 사용자 입력의 정확성을 높이고 UX를 개선하는 데 필수적이다.

8.10 HTML5 폼 검증 속성 및 사용자 정의 메시지

HTML5에서는 **자바스크립트 없이도 폼 입력을 검증**할 수 있도록 다양한 속성을 지원한다.
또한, 사용자에게 **직관적인 피드백**을 줄 수 있도록 **사용자 정의 오류 메시지**를 설정하는 방법도 제공된다.

✅ HTML5 기본 폼 검증 속성

속성	설명
`required`	입력 필수 여부
`min`, `max`	숫자, 날짜, 범위 제한
`maxlength`	최대 문자 수 제한

속성	설명
`pattern`	정규 표현식 기반 입력 제한
`type`	입력 형식 (`email`, `url`, `tel`, `number`, `date` 등)
`step`	숫자/날짜 증가 간격 제한

✅ 1. `pattern`: 정규 표현식으로 입력 형식 제한

```
<input type="text" name="userid" pattern="[a-zA-Z0-9]{4,12}" required>
```

- 영문/숫자 4~12자만 허용
- 정규식이 맞지 않으면 브라우저가 폼 제출을 막음

📌 `title` 속성을 함께 사용하면 오류 메시지에 힌트 제공 가능

```
<input pattern="\d{4}" title="4자리 숫자만 입력하세요.">
```

✅ 2. 이메일/URL 유효성 검사 (`type` 속성)

```
<input type="email" required>
<input type="url" required>
```

- 형식이 틀리면 브라우저에서 **기본 오류 메시지** 출력

✅ 3. `maxlength`, `minlength`

```
<input type="text" maxlength="10" minlength="5" required>
```

- 최소 5자, 최대 10자 입력만 허용

✅ 4. `novalidate`: HTML5 검증 비활성화

```
<form novalidate>
```

- 브라우저 기본 검증 기능을 **끄고 싶을 때 사용**

✅ 5. 사용자 정의 오류 메시지 (`setCustomValidity()`)

자바스크립트를 사용해 **브라우저 기본 메시지를 덮어쓰기** 가능

```
<form id="myForm">
  <input id="phone" type="tel" pattern="\d{3}-\d{4}-\d{4}" required>
  <button type="submit">제출</button>
</form>

<script>
  const phone = document.getElementById("phone");

  phone.addEventListener("input", () => {
    if (phone.validity.patternMismatch) {
      phone.setCustomValidity("전화번호는 000-0000-0000 형식이어야 합니다.");
    } else {
      phone.setCustomValidity(""); // 정상화
    }
  });
</script>
```

📌 `setCustomValidity('')`로 메시지를 지우지 않으면, 입력이 올바르더라도 제출 불가 상태가 유지됨

✅ 6. `checkValidity()` / `reportValidity()`

- `checkValidity()`: 유효성 검사 결과를 boolean으로 반환
- `reportValidity()`: 오류 메시지를 **브라우저에 시각적으로 표시**

```
const form = document.getElementById("myForm");
if (!form.checkValidity()) {
  form.reportValidity(); // 오류 표시
}
```

✅ 예제 통합

```
<form onsubmit="return validateForm()" novalidate>
  <label>이메일:</label>
  <input id="email" type="email" required>
  <button type="submit">제출</button>
</form>

<script>
function validateForm() {
  const email = document.getElementById("email");
  if (!email.checkValidity()) {
    email.setCustomValidity("올바른 이메일 주소를 입력해주세요.");
    email.reportValidity();
    return false;
```

```
  }
  return true;
}
</script>
```

✅ 한 줄 요약

> HTML5는 `required`, `pattern`, `type`, `maxlength` 등의 속성을 통해 자바스크립트 없이도 강력한 폼 검증을 제공하며, 필요 시 `setCustomValidity()`로 사용자 맞춤 메시지를 설정할 수 있다.