19. 실습/응용

19.1 개인 블로그 마크업 작성

✅ 목표

개인 블로그를 HTML로 직접 마크업하며, 시맨틱 태그를 활용해 **구조적이고 접근성 높은 HTML 문서**를 작성해보자.
이 예제는 다음을 포함한다:

- 시맨틱 구조 (`<header>`, `<main>`, `<article>`, `<section>`, `<footer>`)
- 내비게이션 메뉴
- 블로그 글 목록
- 포스트 상세 구조
- 반응형 지원을 위한 viewport 설정

📜 전체 HTML 예제

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>정석의 블로그</title>
  <link rel="stylesheet" href="styles.css">
  <meta name="description" content="개발과 공부를 위한 정석의 기술 블로그입니다.">
</head>
<body>
  <header>
    <h1><a href="/">정석의 블로그</a></h1>
    <nav>
      <ul>
        <li><a href="/about.html">소개</a></li>
        <li><a href="/posts.html">포스트</a></li>
        <li><a href="/contact.html">연락처</a></li>
      </ul>
    </nav>
  </header>

  <main>
    <section aria-labelledby="latest-posts">
      <h2 id="latest-posts">최신 포스트</h2>

      <article>
        <h3><a href="/posts/hello-world.html">블로그에 오신 걸 환영합니다</a></h3>
        <p>처음으로 글을 씁니다. 앞으로 개발과 학습 경험을 공유할 예정입니다.</p>
        <p><time datetime="2025-05-29">2025년 5월 29일</time></p>
      </article>

```

```
      <article>
        <h3><a href="/posts/html-semantics.html">HTML 시맨틱 태그 완전 정복</a></h3>
        <p>웹의 구조를 명확하게 표현하기 위한 시맨틱 태그 정리.</p>
        <p><time datetime="2025-05-20">2025년 5월 20일</time></p>
      </article>

    </section>
  </main>

  <footer>
    <p>© 2025 정석. 모든 권리 보유.</p>
    <p><a href="/privacy.html">개인정보 처리방침</a></p>
  </footer>
</body>
</html>
```

📌 구조 요약

영역	사용 태그	설명
헤더	`<header>`, `<nav>`, `<ul>`	사이트 제목 + 메뉴
메인 콘텐츠	`<main>`, `<section>`, `<article>`	포스트 목록 구성
포스트	`<article>`, `<h3>`, `<time>`	개별 글 정보
푸터	`<footer>`	저작권 및 링크

✅ 시맨틱 요소 의미

태그	의미
`<header>`	문서나 섹션의 헤더
`<nav>`	주요 내비게이션 링크 묶음
`<main>`	페이지의 핵심 콘텐츠
`<article>`	독립적으로 배포 가능한 콘텐츠 단위 (블로그 글 등)
`<section>`	주제별 콘텐츠 구간
`<footer>`	문서나 섹션의 바닥글 정보
`<time>`	날짜/시간 정보 (검색엔진 친화적)

🧠 확장 가능 기능 (다음 실습에 활용 가능)

- 개별 포스트 페이지 마크업 (`/posts/slug.html`)
- 태그/카테고리 분류
- 검색 기능 마크업
- 댓글 영역 (`<form>` 활용)
- PWA 설치 가능 블로그로 확장

💡 참고

- 디자인은 CSS에서 적용 (다음 단계에서 `styles.css` 제공 가능)
- 추후 Markdown → HTML 변환을 위한 구조로도 활용 가능
- SEO 및 접근성을 위한 `<meta>`, `<time>`, `aria-*` 포함

19.2 HTML 기반 포트폴리오 페이지 만들기

✅ 목표

HTML만으로 **개인 포트폴리오 페이지의 마크업 구조**를 설계해보자.
이 예제는 개발자/디자이너/학생 등의 자기소개, 프로젝트, 기술 스택, 연락처 등을 시맨틱하게 구성하며, 다음을 포함한다:

- 시맨틱 구조 (`<header>`, `<section>`, `<footer>`)
- 소개, 기술 스택, 프로젝트 목록, 연락처
- 구조화된 정보 전달
- 스타일은 분리(`styles.css`)하고 마크업에 집중

📜 전체 HTML 예제

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>정석의 포트폴리오</title>
  <link rel="stylesheet" href="styles.css" />
  <meta name="description" content="정석의 개발자 포트폴리오. 프로젝트, 기술 스택, 연락처 제공." />
</head>
<body>
  <header>
    <h1>정석</h1>
    <p>풀스택 개발자 · 문제 해결러 · 창작을 좋아하는 기술인</p>
    <nav>
      <ul>
        <li><a href="#about">소개</a></li>
        <li><a href="#skills">기술 스택</a></li>
```

```
        <li><a href="#projects">프로젝트</a></li>
        <li><a href="#contact">연락처</a></li>
      </ul>
    </nav>
  </header>

  <main>
    <section id="about">
      <h2>👤 소개</h2>
      <p>안녕하세요! 저는 사용자 중심의 웹 애플리케이션을 만드는 데 열정을 가진 풀스택 개발자 정석입니다. Spring, Django, React, ROS2 등 다양한 기술 스택을 활용하며, 제품을 끝까지 책임지는 엔지니어링을 추구합니다.</p>
    </section>

    <section id="skills">
      <h2>🛠 기술 스택</h2>
      <ul>
        <li><strong>프론트엔드:</strong> HTML, CSS, JavaScript, React</li>
        <li><strong>백엔드:</strong> Java (Spring Boot), Python (Django), Node.js</li>
        <li><strong>데이터베이스:</strong> MariaDB, PostgreSQL, MongoDB</li>
        <li><strong>기타:</strong> Git, Docker, Linux, ROS2</li>
      </ul>
    </section>

    <section id="projects">
      <h2>🚀 프로젝트</h2>

      <article>
        <h3>📦 큐브위성 지상국 모니터링 시스템</h3>
        <p>NASA cFS 기반의 OBC와 연동되는 웹 기반 실시간 지상국 GUI 개발. Spring Boot + WebSocket + Chart.js 활용</p>
        <ul>
          <li>실시간 Telemetry 수신</li>
          <li>명령어 전송 및 상태 시각화</li>
          <li>ROS2 ↔ Spring 연동 구조 설계</li>
        </ul>
      </article>

      <article>
        <h3>📝 토익 단어 암기 웹앱</h3>
        <p>HTML/CSS/JS 기반 SPA 구조로 제작한 온라인 단어 학습 웹앱</p>
        <ul>
          <li>IndexedDB 기반 클라이언트 저장</li>
          <li>드래그 기반 테스트 인터페이스</li>
        </ul>
      </article>
    </section>

    <section id="contact">
      <h2>📬 연락처</h2>
      <p>이메일: <a href="mailto:jeongseok.dev@example.com">jeongseok.dev@example.com</a></p>
      <p>GitHub: <a href="https://github.com/jeongseok">github.com/jeongseok</a></p>
```

```
      <p>LinkedIn: <a
href="https://linkedin.com/in/jeongseok">linkedin.com/in/jeongseok</a></p>
    </section>
  </main>

  <footer>
    <p>© 2025 정석. 모든 권리 보유.</p>
  </footer>
</body>
</html>
```

📌 구조 요약

섹션 ID	내용 구성
`#about`	자기소개
`#skills`	기술 스택
`#projects`	프로젝트 이력 및 설명
`#contact`	이메일, GitHub, LinkedIn
`<nav>`	상단 메뉴로 각 섹션 이동

✅ 시맨틱 구조 포인트

태그	의미
`<section>`	개별 기능/콘텐츠 블록
`<article>`	독립적인 프로젝트 하나
`<ul>`, `<li>`	기술 나열, 기능 설명
`<a>`	외부 링크, 이메일 링크
`<strong>`	강조 텍스트 (접근성 유지)

💡 확장 아이디어

- 다국어 지원 (`lang`, `hreflang`, `<html lang="en">`)
- 다크 모드 토글
- 애니메이션/스크롤 인터랙션 추가
- HTML + PWA로 오프라인 포트폴리오
- React/Next 기반으로 변환 후 배포

19.3 폼 입력 → 서버 전송 구조 구현

✅ 목표

사용자가 폼에 입력한 데이터를 **HTML + `<form>` 태그로 수집하고**, 서버로 전송하는 기본 구조를 구현한다.
이 구조는 로그인, 회원가입, 문의하기, 댓글 작성 등의 웹 애플리케이션에 활용된다.

📁 1. 기본 폼 HTML 예제

```html
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>문의하기</title>
</head>
<body>
  <h1>📧 문의하기</h1>

  <form action="/submit" method="POST">
    <label for="name">이름:</label><br>
    <input type="text" id="name" name="name" required><br><br>

    <label for="email">이메일:</label><br>
    <input type="email" id="email" name="email" required><br><br>

    <label for="message">메시지:</label><br>
    <textarea id="message" name="message" rows="5" required></textarea><br><br>

    <button type="submit">보내기</button>
  </form>
</body>
</html>
```

📌 핵심 속성 정리

속성	설명
`action="/submit"`	데이터를 보낼 서버의 URL 경로
`method="POST"`	POST 방식으로 전송 (보안 및 양 많을 때 사용)
`name="..."`	폼 필드의 이름 (서버에서 key로 사용됨)
`required`	필수 입력 필드로 지정

🛠️ 2. 서버 측 처리 예제 (Node.js + Express 기준)

```
// server.js
const express = require('express');
const app = express();
const port = 3000;

// POST 데이터를 파싱하기 위한 미들웨어
app.use(express.urlencoded({ extended: true }));
app.use(express.json());

// 요청 처리
app.post('/submit', (req, res) => {
  const { name, email, message } = req.body;
  console.log('폼 데이터 수신됨:', { name, email, message });

  // 여기서 DB 저장, 이메일 전송 등 처리 가능
  res.send('문의 감사합니다! 빠르게 연락드리겠습니다.');
});

app.listen(port, () => {
  console.log(`✅ 서버 실행됨: http://localhost:${port}`);
});
```

> 📦 위 코드는 Express.js 서버에서 폼 데이터를 처리하는 기본 구조이다.

🧪 3. 전송 구조 확인 (데이터 흐름)

```
[사용자 입력]
   ↓
<form> 태그로 입력 수집
   ↓
POST 요청 (Content-Type: application/x-www-form-urlencoded)
   ↓
서버의 /submit 라우트에서 req.body로 수신
   ↓
DB 저장 / 메일 발송 / 응답 메시지 반환
```

🔐 4. 실전 적용 시 고려 사항

항목	설명
입력값 검증	클라이언트뿐 아니라 서버에서도 유효성 검사 필수
XSS/Injection 방지	HTML 태그 escape, DB 쿼리 escape 필수
CSRF 보호	토큰 기반 보호(`csrfToken`, `SameSite` 설정 등)
CORS 설정	클라이언트와 서버가 다른 origin이면 헤더 필요

✅ 확장 기능 (향후 예제에서 다룸)

- AJAX 기반 비동기 전송 (`fetch()` 활용)
- 파일 업로드 (`<input type="file">`)
- 백엔드 언어별 연동: Python(Django/Flask), PHP, Java(Spring) 등
- FormData 객체 사용

19.4 HTML 이메일 템플릿 작성

✅ 목표

HTML 기반으로 **디자인된 이메일 콘텐츠를 만들기 위한 템플릿 마크업 구조**를 작성한다.
이메일은 웹과 달리 **렌더링 환경이 매우 제한적**이기 때문에, **인라인 스타일, 테이블 레이아웃, 고정 폭** 등의 방식으로 작성해야 한다.

> 💡 **이메일 클라이언트(Outlook, Gmail, Apple Mail 등)**은 CSS 지원이 불완전하기 때문에 반드시 **최소한의, 안정적인 HTML**을 사용해야 한다.

📁 기본 HTML 이메일 템플릿 예제

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <title>이메일 템플릿</title>
</head>
<body style="margin:0; padding:0; font-family:Arial, sans-serif; background-color:#f4f4f4;">

  <table width="100%" cellpadding="0" cellspacing="0" border="0" bgcolor="#f4f4f4">
    <tr>
      <td align="center">
        <!-- ✉ 메인 컨테이너 테이블 -->
        <table width="600" cellpadding="0" cellspacing="0" border="0" bgcolor="#ffffff" style="margin:20px auto; padding:20px; border-radius:6px; box-shadow:0 0 5px rgba(0,0,0,0.1);">
          <!-- 로고 -->
          <tr>
            <td align="center" style="padding:20px 0;">
              <img src="https://example.com/logo.png" alt="회사 로고" width="120" style="display:block;">
            </td>
          </tr>

          <!-- 제목 -->
          <tr>
            <td align="center" style="padding:10px 30px; font-size:24px; font-weight:bold; color:#333333;">
              정석님의 문의가 접수되었습니다!
```

```
            </td>
          </tr>

          <!-- 본문 내용 -->
          <tr>
            <td style="padding:20px 30px; font-size:16px; color:#555555; line-height:1.6;">
              안녕하세요 정석님,<br><br>
              귀하의 문의를 잘 접수하였습니다. 담당자가 빠른 시일 내에 회신드릴 예정입니다.<br><br>
              문의 내용:<br>
              <em>"HTML 이메일 템플릿 관련 내용을 요청하였습니다."</em><br><br>

              감사합니다.<br>
              <strong>정석기술지원팀</strong>
            </td>
          </tr>

          <!-- 버튼 -->
          <tr>
            <td align="center" style="padding:20px;">
              <a href="https://example.com/support" target="_blank"
                 style="display:inline-block; padding:12px 24px; background-color:#007BFF; color:#ffffff; text-decoration:none; border-radius:4px; font-weight:bold;">
                지원 센터로 이동
              </a>
            </td>
          </tr>

          <!-- 푸터 -->
          <tr>
            <td align="center" style="font-size:12px; color:#999999; padding:20px 30px;">
              본 메일은 noreply@example.com에서 자동 발송되었습니다.<br>
              서울특별시 테크로 123, 정석빌딩 5층 | 구독취소: <a href="#" style="color:#999999;">Unsubscribe</a>
            </td>
          </tr>
        </table>
        <!-- / 메인 컨테이너 테이블 -->
      </td>
    </tr>
  </table>

</body>
</html>
```

🧠 주요 구성 포인트

항목	작성 방식
전체 레이아웃	`<table>` 기반으로 구성
폰트, 색상, 여백	**모든 스타일을 인라인**으로 적용
버튼	`<a>`에 인라인 스타일 적용
이미지	반드시 `alt` 속성과 고정 `width` 지정
반응형	매우 제한적 (미디어쿼리 거의 미지원)

❗ 이메일 마크업 작성 시 주의사항

- **CSS 클래스 사용 금지** (대부분 클라이언트에서 무시됨)
- **외부 CSS 링크 불가**
- **폼, JS, 영상 등 비지원**
- **폰트는 웹 안전 폰트(Arial, Verdana, Georgia 등)** 사용
- **이미지는 절대경로(URL)**로 지정

📩 적용 시나리오 예시

- 문의 접수 확인 메일
- 뉴스레터 발송
- 결제 영수증
- 이벤트 알림

🧪 테스트 도구 추천

- Litmus – 클라이언트별 이메일 테스트
- Email on Acid
- Mailtrap.io – 개발 환경 테스트용 이메일 서버

✅ 정리

항목	설명
마크업 방식	`<table>` 중심, inline 스타일
지원 클라이언트	Gmail, Outlook, Apple Mail 등
접근성 요소	`alt`, 구조적 텍스트

항목	설명
반응형/동적 콘텐츠	매우 제한적 (SPA 불가)

19.5 HTML 기반 UI 디자인(모달, 탭, 아코디언)

✅ 목표

JavaScript 없이 또는 최소한의 JS로 구현 가능한 **모달, 탭, 아코디언** UI 컴포넌트를 HTML 구조 중심으로 설계한다.
접근성(ARIA) 및 시맨틱 구조도 고려하며, CSS-only 버전도 포함.

📦 1. 모달 (Modal Dialog)

📜 기본 HTML + CSS 구조

```
<!-- 모달 열기 버튼 -->
<button id="openModal">모달 열기</button>

<!-- 모달 영역 -->
<div id="modal" class="modal">
  <div class="modal-content" role="dialog" aria-modal="true" aria-labelledby="modalTitle">
    <h2 id="modalTitle">모달 제목</h2>
    <p>이것은 모달 내용입니다.</p>
    <button id="closeModal">닫기</button>
  </div>
</div>
```

🖼️ CSS

```
.modal {
  display: none;
  position: fixed; top: 0; left: 0; width: 100%; height: 100%;
  background-color: rgba(0,0,0,0.5);
  justify-content: center; align-items: center;
}
.modal-content {
  background: white; padding: 20px; border-radius: 5px;
}
.modal.show {
  display: flex;
}
```

🔧 JS (기본)

```
document.getElementById('openModal').onclick = () =>
  document.getElementById('modal').classList.add('show');

document.getElementById('closeModal').onclick = () =>
  document.getElementById('modal').classList.remove('show');
```

📦 2. 탭 (Tab Panel)

📜 HTML 구조

```
<div class="tabs">
  <ul role="tablist">
    <li><button role="tab" aria-controls="tab1" aria-selected="true">탭 1</button></li>
    <li><button role="tab" aria-controls="tab2">탭 2</button></li>
  </ul>

  <div id="tab1" role="tabpanel">탭 1의 내용입니다.</div>
  <div id="tab2" role="tabpanel" hidden>탭 2의 내용입니다.</div>
</div>
```

🔧 JS (간단 탭 스위칭)

```
const tabs = document.querySelectorAll('[role="tab"]');
tabs.forEach(tab => {
  tab.addEventListener('click', () => {
    tabs.forEach(t => t.setAttribute('aria-selected', false));
    tab.setAttribute('aria-selected', true);

    document.querySelectorAll('[role="tabpanel"]').forEach(panel =>
      panel.hidden = true);
    document.getElementById(tab.getAttribute('aria-controls')).hidden = false;
  });
});
```

📦 3. 아코디언 (Accordion)

📜 HTML + `<details>` 기반

```
<details>
  <summary>자주 묻는 질문 1</summary>
  <p>답변 내용입니다. 접히고 펼쳐집니다.</p>
</details>

<details>
  <summary>자주 묻는 질문 2</summary>
  <p>이 내용도 접이식입니다.</p>
```

```
</details>
```

✅ 장점

- JS 없이 기본 접기/펼치기 가능
- 브라우저 기본 스타일 + 접근성 지원

🧠 정리: UI 컴포넌트 요약

컴포넌트	핵심 HTML 태그	JS 필요 여부	접근성 고려 포인트
모달	`div`, `role=dialog`	✔ 필요	`aria-modal`, `aria-labelledby`
탭	`button`, `div`, `role=tab`	✔ 필요	`aria-controls`, `aria-selected`
아코디언	`<details>`, `<summary>`	❌ 불필요	기본 제공 (ARIA 포함됨)

💡 활용 예

- 모달: 사용자 입력/확인 창
- 탭: 마이페이지, 설정, 탭 전환 UI
- 아코디언: FAQ, 설정 확장 정보