1시간**:** 웹 및 **HTML** 기본 소개

이 시간은 웹 개발의 세계로 들어가는 첫 걸음입니다. 웹 개발이란 무엇인지, 클라이언트-서버 모델, 프런트엔드와 백엔드의 차이점을 소개합니다. 이어서 웹 콘텐츠 구조화에 필수적인 HTML(HyperText Markup Language)의 역할을 다룹니다. 학습 환경 설정은 이 과정의 중요한 초기 단계입니다. Visual Studio Code 설치와 Live Server 확장 프로그램 설정은 학습자가 코드를 작성하고 즉시 결과를 확인할 수 있도록 돕습니다. 이러한 초기 환경 설정이 원활하게 이루어지는 것은 초보 학습자의 학습 참여와 자신감 형성에 결정적인 영향을 미칩니다. 많은 학습자가 초기 기술적 문제로 인해 좌절하고 이탈하는 경향이 있기 때문에, 이 단계에서 명확하고 단계별 지침을 제공하는 것이 중요합니다. 웹 브라우저가 페이지를 요청하고 표시하는 방식, VS Code와 같은 통합 개발 환경(IDE)의 목적, 그리고 <!DOCTYPE html>, <html>, <head>, <body>와 같은 HTML 문서의 기본 구조를 이해하는 것이 핵심 개념입니다. 첫 번째 index.html 파일을 만들고, 기본적인 "Hello World" 페이지를 작성하며, Live Server를 사용하여 브라우저에서 렌더링되는 것을 관찰하는 실습을 진행합니다.

핵심 개념**:**

- 웹 개발이란**?**
- 클라이언트**-**서버 모델
- 프런트엔드 vs 백엔드
- **HTML**의 역할
- **VS Code** 및 **Live Server** 환경 설정
- 웹 브라우저의 작동 방식
- **HTML** 문서의 기본 구조 (<!DOCTYPE html>, <html>, <head>, <body>)

실습 예제**:**

1. **VS Code** 설치 및 **Live Server** 확장 프로그램 설정**:**
   - VS Code 공식 웹사이트에서 다운로드 및 설치합니다.
   - VS Code를 실행한 후, 확장(Extensions) 탭에서 "Live Server"를 검색하여 설치합니다.
2. 첫 번째 **index.html** 파일 생성**:**
   - VS Code에서 새 폴더를 만들고, 해당 폴더 안에 index.html 파일을 생성합니다.
3. **"Hello World"** 페이지 작성**:**
   - index.html 파일에 다음 코드를 작성합니다.

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>나의 첫 웹페이지</title>
```

```
</head>
<body>
  <h1>안녕하세요, 웹 개발!</h1>
  <p>이것은 나의 첫 번째 HTML 페이지입니다.</p>
</body>
</html>
```

4. **Live Server**로 브라우저에서 확인**:**
   - index.html 파일에서 마우스 오른쪽 버튼을 클릭하여 "Open with Live Server"를 선택하거나, VS Code 하단의 "Go Live" 버튼을 클릭합니다.
   - 웹 브라우저에서 "안녕하세요, 웹 개발!" 텍스트와 "이것은 나의 첫 번째 HTML 페이지입니다." 단락이 표시되는 것을 확인합니다.

2시간**:** 필수 **HTML** 태그를 통한 콘텐츠 구조화

이 시간에는 콘텐츠의 계층적 구조를 위한 제목(<h1>부터 <h6>), 텍스트 블록을 위한 단락(<p>), 강제 줄 바꿈을 위한 줄 바꿈(
), 주제별 구분을 위한 수평선(<hr>), 그리고 기본적인 텍스트 서식 태그((중요), (강조), (인라인 그룹화), (블록 레벨 그룹화))를 다룹니다. 블록 레벨 요소와 인라인 요소의 차이점을 이해하고, 시맨틱 태그와 비시맨틱 태그의 개념을 처음으로 소개하는 것이 중요합니다. 간단한 블로그 게시물이나 기사를 구조화하고, 적절한 제목과 단락을 적용하며, 텍스트 서식 태그를 사용하여 특정 단어나 구문을 강조하는 실습을 진행합니다.

핵심 개념**:**

- 제목 태그 (<h1> ~ <h6>)
- 단락 태그 (<p>)
- 줄 바꿈 태그 (
)
- 수평선 태그 (<hr>)
- 텍스트 서식 태그 (, ,)
- 블록 레벨 그룹화 태그 ( )
- 블록 레벨 요소와 인라인 요소의 차이점
- 시맨틱 태그와 비시맨틱 태그 소개

실습 예제**:**

1. 새로운 **HTML** 파일 생성**:**
   - content-structure.html 파일을 생성하고 다음 코드를 작성합니다.

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

```
    <title>콘텐츠 구조화 예제</title>
</head>
<body>
    <h1>주요 제목: 나의 웹 개발 여정</h1>
    <p>이것은 웹 개발 학습의 <strong>첫 번째 단계</strong>입니다. 우리는 HTML의
<em>기본적인 요소들</em>을 학습하고 있습니다.</p>

    <h2>HTML의 중요성</h2>
    <p>HTML은 웹 페이지의 뼈대를 만듭니다. 모든 웹 콘텐츠는 이 구조 위에 놓이게
됩니다.</p>
    <p>새로운 줄을 사용하고 싶을 때는 <br> 이렇게 `<br>` 태그를 사용할 수
있습니다.</p>
    <hr>

    <h3>블록 요소와 인라인 요소</h3>
    <p>
        <div style="border: 1px solid blue; padding: 10px; margin: 5px;">
            이것은 `<div>` 태그로 묶인 블록 레벨 요소입니다. <span
style="background-color: yellow;">`<span>`은 인라인 요소입니다.</span>
        </div>
    </p>
    <p>또 다른 단락입니다. 여기에는 <strong>매우 중요한 정보</strong>가 담겨
있습니다.</p>
    <p>
        <span style="color: red;">이 텍스트는 인라인으로 스타일링된
텍스트입니다.</span>
        <span style="font-style: italic;">이 텍스트는 기울임꼴입니다.</span>
    </p>
</body>
</html>
```

2. 브라우저에서 확인**:**
    - Live Server로 content-structure.html 파일을 열어 다양한 제목, 단락, 줄 바꿈, 수평선, 그리고 강조된 텍스트가 어떻게 표시되는지 확인합니다.
    - 와 에 적용된 간단한 인라인 스타일을 통해 블록/인라인 요소의 차이를 시각적으로 확인합니다.

3시간**:** 링크**,** 이미지 및 목록

이 시간은 웹 페이지의 상호 연결성과 시각적 요소를 다룹니다. 탐색을 위한 하이퍼링크(<a> 태그), 시각적 콘텐츠를 위한 이미지 삽입(<img> 태그), 번호 매겨진 순서를 위한 순서 있는 목록(<ol>, <li>), 그리고 글머리 기호 항목을 위한 순서 없는 목록(<ul>, <li>)을 학습합니다. 링크와 이미지에 대한 상대 및 절대 파일 경로를 이해하고, 기능과 접근성을 위한 속성(href, src, alt)의 중요성을 파악하는 것이 핵심 개념입니다.

내부 및 외부 탐색 링크를 만들고, 대체 텍스트와 함께 이미지를 페이지에 추가하며, 레시피 목록이나 간단한 목차와 같은 구조화된 목록을 만드는 실습을 진행합니다.

핵심 개념**:**

- 하이퍼링크 태그 (<a>): href 속성
- 이미지 태그 (<img>): src, alt, width, height 속성
- 순서 있는 목록 (<ol>, <li>)
- 순서 없는 목록 (<ul>, <li>)
- 상대 경로 vs 절대 경로
- 접근성을 위한 alt 속성의 중요성

실습 예제**:**

1. 새로운 **HTML** 파일 생성**:**
    - links-images-lists.html 파일을 생성하고 다음 코드를 작성합니다.

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>링크, 이미지, 목록 예제</title>
</head>
<body>
  <h1>웹 페이지 상호작용</h1>

  <h2>유용한 링크</h2>
  <ul>
    <li><a href="https://www.google.com" target="_blank">구글 방문하기 (새 탭)</a></li>
    <li><a href="https://developer.mozilla.org/ko/docs/Web/HTML" target="_blank">MDN HTML
문서 (새 탭)</a></li>
    <li><a href="about.html">회사 소개 페이지로 이동 (상대 경로 예시)</a></li>
    <li><a href="#bottom-section">페이지 하단으로 이동</a></li>
  </ul>

  <h2>갤러리</h2>
  <!-- 예제 이미지가 없으면, 온라인에서 적절한 이미지를 검색하여 사용하거나 placeholder.com과
같은 서비스 이용 -->
  <img src="https://via.placeholder.com/300x200" alt="아름다운 풍경 사진" width="300"
height="200">
  <p>설명: 이 이미지는 웹 페이지의 시각적 요소를 보여줍니다.</p>

  <img src="images/local-example.jpg" alt="로컬 폴더에 있는 이미지 (경로를 맞춰주세요)"
width="250" height="150">
  <p>설명: 이 이미지는 `images` 폴더에 있는 로컬 파일입니다.</p>
  <hr>

  <h2>할 일 목록</h2>
  <ol>
    <li>HTML 학습하기</li>
```

```
    <li>CSS 학습하기</li>
    <li>JavaScript 학습하기</li>
    <li>미니 프로젝트 아이디어 구상</li>
  </ol>

  <h2>좋아하는 과일 목록</h2>
  <ul>
    <li>사과</li>
    <li>바나나
      <ul>
        <li>몽키 바나나</li>
        <li>필리핀 바나나</li>
      </ul>
    </li>
    <li>오렌지</li>
    <li>딸기</li>
  </ul>

  <p id="bottom-section" style="margin-top: 500px;">이곳은 페이지의 하단입니다.</p>
</body>
</html>
```

2. **about.html** 파일 **(**상대 경로 예시용**)** 생성**:**
   - about.html 파일을 생성하고 다음 코드를 작성합니다.

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>회사 소개</title>
</head>
<body>
  <h1>회사 소개 페이지</h1>
  <p>저희 회사는 혁신적인 웹 솔루션을 제공합니다.</p>
  <p><a href="links-images-lists.html">이전 페이지로 돌아가기</a></p>
</body>
</html>
```

3. 이미지 폴더 및 파일 준비 **(**선택 사항**):**
   - links-images-lists.html 파일과 같은 위치에 images 폴더를 만들고, 그 안에 local-example.jpg라는 이미지 파일을 넣습니다. (없으면 온라인 이미지 URL을 사용하거나 이 부분을 생략해도 됩니다.)
4. 브라우저에서 확인**:**
   - Live Server로 links-images-lists.html 파일을 열어 외부 링크, 내부 링크, 이미지, 그리고 다양한 목록들이 어떻게 작동하는지 확인합니다.
   - 링크를 클릭하여 페이지 이동과 새 탭 열림을 확인하고, 이미지와 목록이 잘

표시되는지 확인합니다.

4시간**:** 폼**,** 테이블 및 시맨틱 **HTML**

이 시간은 사용자 입력과 표 형식 데이터 표현에 초점을 맞춥니다. 사용자 입력을 위한 기본적인 폼 요소 (<form>, <input type="text/password/submit/checkbox/radio">, <button>), 표 형식 데이터를 위한 테이블 (<table>, <thead>, <tbody>, <tr>, <th>, <td>)을 소개합니다. 또한, 시맨틱 HTML5 태그(<thead>, <nav>, <main>, <section>, <footer>, <article>, <aside>)의 중요성을 더 깊이 다룹니다. 폼 제출의 이해, 테이블의 구조화된 특성, 그리고 웹 접근성, 검색 엔진 최적화(SEO), 유지보수 가능한 코드를 위한 시맨틱 태그의 중요한 역할이 핵심 개념입니다. 다양한 입력 유형과 제출 버튼이 있는 간단한 연락처 폼을 만들고, 기본적인 데이터 테이블(예: 수업 시간표)을 구축하며, 이전에 만든 페이지를 적절한 시맨틱 HTML5 태그를 포함하도록 리팩토링하는 실습을 진행합니다.

시맨틱 **HTML**의 중요성은 아무리 강조해도 지나치지 않습니다. 16시간 과정에서 이 개념의 모든 깊이를 다룰 수는 없지만, <header> 대신 일반적인 를 사용하는 대신 <header>와 같은 시맨틱 태그를 처음부터 도입하는 것은 좋은 개발 습관을 심어주는 데 매우 효과적입니다. 초보자가 나중에 비시맨틱 구조를 학습하고 리팩토링하는 것보다 처음부터 올바른 관행을 배우고 채택하는 것이 훨씬 쉽습니다. 이는 향후 학습과 전문적인 코드 품질을 위한 강력한 기반을 마련합니다. 따라서 커리큘럼은 단순히 HTML 태그를 사용하는 방법을 가르치는 것을 넘어, 시맨틱 태그가 왜 선호되는지, 그리고 화면 판독기 호환성 및 검색 엔진 가시성과 같은 실제 이점과 어떻게 연결되는지 명시적으로 설명해야 합니다.

핵심 개념**:**

- 폼 (<form>) 및 입력 요소 (<input>, <label>, <button>)
  - type 속성: text, password, submit, checkbox, radio
- 테이블 (<table>, <thead>, <tbody>, <tr>, <th>, <td>)
- 시맨틱 **HTML5** 태그: <header>, <nav>, <main>, <section>, <footer>, <article>, <aside>
- 폼 제출의 작동 방식
- 테이블의 구조화된 데이터 표현
- 시맨틱 태그의 웹 접근성, **SEO**, 유지보수성 이점

실습 예제**:**

1. 새로운 **HTML** 파일 생성**:**
   - forms-tables-semantic.html 파일을 생성하고 다음 코드를 작성합니다.

```
<!DOCTYPE html>
<html lang="ko">
<head>
```

```
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>폼, 테이블, 시맨틱 HTML 예제</title>
</head>
<body>
  <header>
    <h1>우리 웹사이트</h1>
    <nav>
      <ul>
        <li><a href="/">홈</a></li>
        <li><a href="/services">서비스</a></li>
        <li><a href="/contact">문의</a></li>
      </ul>
    </nav>
  </header>

  <main>
    <section>
      <h2>문의하기</h2>
      <form action="/submit-form" method="post">
        <label for="name">이름:</label>
        <input type="text" id="name" name="name" required
placeholder="이름을 입력하세요"><br><br>

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

        <label for="password">비밀번호:</label>
        <input type="password" id="password" name="password"
minlength="8"><br><br>

        <p>선호하는 연락 방법:</p>
        <input type="radio" id="email-contact" name="contact-method"
value="email">
        <label for="email-contact">이메일</label><br>
        <input type="radio" id="phone-contact" name="contact-method"
value="phone">
        <label for="phone-contact">전화</label><br><br>

        <input type="checkbox" id="newsletter" name="newsletter"
value="yes">
        <label for="newsletter">뉴스레터 구독</label><br><br>

        <button type="submit">제출하기</button>
```

```
        <button type="reset">초기화</button>
      </form>
    </section>

    <section>
      <h2>제품 목록</h2>
      <table>
        <thead>
          <tr>
            <th>제품명</th>
            <th>가격</th>
            <th>재고</th>
          </tr>
        </thead>
        <tbody>
          <tr>
            <td>노트북</td>
            <td>1,200,000원</td>
            <td>100</td>
          </tr>
          <tr>
            <td>마우스</td>
            <td>25,000원</td>
            <td>500</td>
          </tr>
          <tr>
            <td>키보드</td>
            <td>80,000원</td>
            <td>200</td>
          </tr>
        </tbody>
        <tfoot>
          <tr>
            <td colspan="3">총 3가지 제품</td>
          </tr>
        </tfoot>
      </table>
    </section>

    <article>
      <h3>최신 소식</h3>
      <p>오늘 새로운 기능이 출시되었습니다. 이 기능은 사용자 경험을 크게
향상시킬 것입니다.</p>
      <aside>
        <p>관련 기사: 웹 개발 트렌드</p>
```

```
        </aside>
      </article>
    </main>

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

2. 브라우저에서 확인**:**
    - Live Server로 forms-tables-semantic.html 파일을 열어 폼 요소들을 직접 입력하고, 테이블의 구조를 확인하며, <header>, <nav>, <main>, <section>, <article>, <footer> 등의 시맨틱 태그가 어떻게 페이지의 논리적 구조를 형성하는지 살펴봅니다. (시각적으로 큰 변화는 없지만, 코드의 의미를 파악하는 것이 중요합니다.)

모듈 **2: CSS** 기초 **(4**시간**)**

5시간**: CSS** 소개 및 기본 스타일링

이 시간은 웹 페이지의 시각적 표현을 담당하는 CSS(Cascading Style Sheets)의 역할에 대해 알아봅니다. CSS 구문(선택자, 속성, 값)과 CSS를 적용하는 세 가지 방법(인라인, 내부, 외부 스타일시트)을 학습합니다. HTML은 구조를, CSS는 표현을 담당하는 "관심사의 분리" 원칙과 color, font-size, background-color와 같은 기본적인 스타일링 속성이 핵심 개념입니다. 세 가지 방법을 모두 사용하여 HTML 요소에 CSS를 적용하고, 텍스트 색상 변경, 글꼴 크기 조정, 페이지의 다른 섹션에 배경색 설정 등을 실험하는 실습을 진행합니다.

핵심 개념**:**

- **CSS**의 역할과 중요성
- **CSS** 구문: 선택자, 속성, 값
- **CSS** 적용 방법: 인라인 스타일, 내부 스타일시트 (<style> 태그), 외부 스타일시트 (<link> 태그)
- "관심사의 분리" 원칙
- 기본 **CSS** 속성: color, font-size, background-color, text-align, font-family

실습 예제**:**

1. 새로운 **HTML** 파일 생성**:**
    - css-basics.html 파일을 생성하고 다음 코드를 작성합니다.

```
<!DOCTYPE html>
<html lang="ko">
```

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS 스타일링 예제</title>
    <!-- 외부 스타일시트 연결 -->
    <link rel="stylesheet" href="styles.css">

    <!-- 내부 스타일시트 -->
    <style>
        h1 {
            color: navy; /* 제목 색상 */
            text-align: center; /* 제목 가운데 정렬 */
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
        }

        .internal-styled-div {
            background-color: #f0f0f0; /* 배경색 */
            padding: 15px; /* 내부 여백 */
            border: 1px solid #ccc; /* 테두리 */
            margin-top: 20px;
        }
    </style>
</head>
<body>
    <h1 style="color: purple; font-size: 3em;">인라인 스타일이 적용된 제목</h1>
    <p style="color: red; font-size: 20px;">이것은 인라인 스타일이 적용된
단락입니다.</p>

    <h1>이것은 내부 스타일시트가 적용된 제목입니다.</h1>
    <div class="internal-styled-div">
        <p>이 단락은 내부 스타일시트로 배경색과 패딩이 적용되었습니다.</p>
    </div>

    <p class="external-styled">이것은 외부 스타일시트가 적용된 단락입니다.</p>
    <div id="anotherExternalDiv">
        <p>이 div는 외부 스타일시트의 ID 선택자로 스타일링됩니다.</p>
    </div>
</body>
</html>
```

2. **styles.css** 파일 생성 **(**외부 스타일시트**):**
    - css-basics.html 파일과 같은 위치에 styles.css 파일을 생성하고 다음 코드를 작성합니다.

```
/* styles.css */
```

```
.external-styled {
  background-color: lightblue;
  font-family: Arial, sans-serif;
  padding: 10px;
  border-radius: 5px;
}

#anotherExternalDiv {
  background-color: lightgoldenrodyellow;
  border: 2px dashed green;
  margin-top: 20px;
  padding: 15px;
  font-size: 1.2em;
}
```

3. 브라우저에서 확인**:**
    - Live Server로 css-basics.html 파일을 열어 인라인, 내부, 외부 스타일시트가 어떻게 HTML 요소에 다양한 시각적 효과를 주는지 확인합니다. 각 스타일 적용 방식의 우선순위(캐스케이딩)에 대해서도 간략히 설명합니다. (인라인 > 내부 > 외부)

6시간**: CSS** 선택자 및 박스 모델

이 시간에는 CSS 선택자(Type, Class, ID), 전체 선택자(*), 그룹화 선택자에 대해 더 깊이 파고듭니다. 또한, CSS 박스 모델(콘텐츠, 패딩, 테두리, 마진)과 그 적용에 대한 포괄적인 이해를 다룹니다. CSS 특이성(규칙의 우선순위), 상속(스타일이 전달되는 방식), 그리고 중요한 box-sizing 속성(border-box)이 핵심 개념입니다. 다양한 선택자 유형을 사용하여 여러 요소를 스타일링하고, 패딩, 테두리, 마진을 적용하여 시각적 간격을 만들고 요소 경계를 정의하며, box-sizing의 효과를 시연하는 실습을 진행합니다.

핵심 개념**:**

- **CSS** 선택자:
    - 타입 선택자 (요소 선택자)
    - 클래스 선택자 (.)
    - **ID** 선택자 (#)
    - 전체 선택자 (*)
    - 그룹화 선택자 (,)
- **CSS** 박스 모델: 콘텐츠, 패딩, 테두리, 마진
    - padding
    - border
    - margin
- box-sizing 속성: content-box (기본값) vs border-box

- **CSS** 특이성 (Specificity)
- **CSS** 상속 (Inheritance)

실습 예제**:**

1. 새로운 **HTML** 파일 생성**:**
   - selectors-box-model.html 파일을 생성하고 다음 코드를 작성합니다.

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>CSS 선택자 및 박스 모델 예제</title>
  <style>
    /* 전체 선택자: 모든 요소에 기본 마진과 패딩 제거 (초기화) */
    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box; /* 모든 요소에 border-box 적용 */
    }

    body {
      font-family: Arial, sans-serif;
      padding: 20px;
    }

    /* 타입 선택자 */
    p {
      color: #333;
      line-height: 1.6;
    }

    h2 {
      color: navy;
      margin-top: 30px;
      margin-bottom: 15px;
    }

    /* 클래스 선택자 */
    .box {
      background-color: lightgray;
      border: 1px solid black;
      margin-bottom: 20px;
      width: 200px;
      height: 100px;
```

```
    text-align: center;
    line-height: 100px; /* 세로 중앙 정렬 */
}

.highlight {
    background-color: yellow;
    padding: 10px; /* 내부 여백 */
    border: 2px dashed orange; /* 테두리 */
    margin-top: 15px; /* 위쪽 외부 여백 */
    margin-bottom: 15px;
}

/* ID 선택자 */
#unique-item {
    font-weight: bold;
    font-size: 24px;
    color: blue;
    border: 3px solid purple;
    padding: 20px;
    margin: 40px;
    background-color: lightcyan;
}

/* 그룹화 선택자 */
h1, h2, h3 {
    text-decoration: underline;
}

/* Box-sizing 예제 */
.content-box-example {
    width: 100px;
    height: 50px;
    padding: 10px;
    border: 2px solid red;
    margin: 10px;
    background-color: #fdd;
    box-sizing: content-box; /* 기본값 */
    float: left; /* 비교를 위해 나란히 배치 */
}

.border-box-example {
    width: 100px;
    height: 50px;
    padding: 10px;
    border: 2px solid green;
```

```
            margin: 10px;
            background-color: #dfd;
            box-sizing: border-box; /* 너비에 패딩과 테두리 포함 */
            float: left; /* 비교를 위해 나란히 배치 */
        }

        .clearfix::after {
            content: "";
            display: table;
            clear: both;
        }
    </style>
</head>
<body>
    <h1>CSS 선택자 및 박스 모델</h1>

    <h2>타입 선택자 및 그룹화 선택자</h2>
    <p>이것은 타입 선택자 `p`에 의해 스타일링된 일반 단락입니다.</p>

    <h2>클래스 선택자</h2>
    <div class="box">일반 박스</div>
    <p class="highlight">이 단락은 클래스 `highlight`로 강조되어 있습니다.</p>
    <div class="box highlight">이 div도 `box`와 `highlight` 클래스가 함께 적용되었습니다.</div>

    <h2>ID 선택자</h2>
    <p id="unique-item">이것은 고유한 스타일을 가진 단락입니다. ID 선택자는 매우 높은 특이성을 가집니다.</p>

    <h2>박스 모델 및 Box-sizing</h2>
    <p>콘텐츠 박스와 보더 박스의 차이를 확인하세요. 개발자 도구로 요소 검사를 추천합니다.</p>
    <div class="clearfix">
        <div class="content-box-example">Content Box</div>
        <div class="border-box-example">Border Box</div>
    </div>
    <div style="clear: both;"></div> <!-- float 해제 -->

    <div class="box" style="margin-top: 30px; padding: 20px; border: 5px solid blue;">
        이 박스에는 패딩 20px, 테두리 5px, 마진 20px이 적용되어 있습니다.
        (위쪽 마진은 `margin-top`으로 설정됨)
    </div>
</body>
</html>
```

2. 브라우저에서 확인**:**
    - Live Server로 selectors-box-model.html 파일을 열어 다양한 선택자들이 요소에 어떻게 스타일을 적용하는지 확인합니다.
    - 가장 중요한 실습은 **브라우저의 개발자 도구 (F12)**를 열어 "Elements" 탭에서 각 요소를 선택하고 "Computed" 또는 "Box Model" 섹션을 확인하는 것입니다. 이를 통해 각 요소의 콘텐츠, 패딩, 테두리, 마진 값이 시각적으로 어떻게 구성되는지 명확하게 이해할 수 있습니다. content-box-example과 border-box-example의 실제 너비/높이를 비교하여 box-sizing의 효과를 시연합니다.

7시간**:** 위치 지정 및 디스플레이 속성

이 시간은 요소 레이아웃에 영향을 미치는 display 속성(block, inline, inline-block, none)과 요소 배치에 대한 세밀한 제어를 위한 position 속성(static, relative, absolute, fixed)을 다룹니다. 레이어링을 위한 z-index도 소개합니다. 일반적인 문서 흐름과 display 및 position 속성이 이를 어떻게 변경하는지 이해하고, z-index로 요소 스택 순서를 관리하는 것이 핵심 개념입니다. 다양한 display 값을 사용하여 페이지에 요소를 배열하고, position을 사용하여 오버레이 또는 고정 헤더/푸터를 만들고, z-index로 레이어링을 시연하는 실습을 진행합니다.

핵심 개념**:**

- display 속성:
    - block: 항상 새 줄에서 시작, 너비/높이/마진/패딩 조절 가능
    - inline: 같은 줄에 배치, 너비/높이 조절 불가, 수평 마진/패딩만 적용
    - inline-block: 같은 줄에 배치, 너비/높이/마진/패딩 조절 가능
    - none: 요소 숨김 (레이아웃 공간 차지 안 함)
- position 속성:
    - static (기본값): 일반 문서 흐름에 따라 배치
    - relative: 일반 문서 흐름에 따라 배치되지만, top, bottom, left, right를 사용하여 상대적으로 이동 가능 (원래 공간 차지)
    - absolute: 가장 가까운 position: relative, absolute, fixed, sticky 속성을 가진 부모 요소 (containing block)를 기준으로 배치. 부모가 없으면 <body>를 기준으로 배치. 일반 문서 흐름에서 벗어남.
    - fixed: 뷰포트 (브라우저 화면)를 기준으로 배치. 스크롤해도 제자리에 고정. 일반 문서 흐름에서 벗어남.
- z-index: position 속성이 static이 아닌 요소들의 쌓임 순서 제어

실습 예제**:**

1. 새로운 **HTML** 파일 생성**:**

- display-position.html 파일을 생성하고 다음 코드를 작성합니다.

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>CSS 위치 지정 및 디스플레이 예제</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin-bottom: 200px; /* fixed 박스를 위한 공간 확보 */
      height: 1500px; /* 스크롤바를 만들기 위해 */
    }

    .box {
      width: 100px;
      height: 50px;
      background-color: lightgray;
      border: 1px solid black;
      margin: 10px;
      text-align: center;
      line-height: 50px;
      font-weight: bold;
      color: #333;
    }

    /* Display 속성 */
    .display-container {
      border: 2px dashed #999;
      padding: 10px;
      margin-bottom: 30px;
    }

    .block-example {
      display: block; /* 새 줄에서 시작, 너비 100% (부모 기준) */
      background-color: lightblue;
    }

    .inline-example {
      display: inline; /* 같은 줄에 배치, 너비/높이 적용 안됨 */
      background-color: lightgreen;
      padding: 5px; /* 패딩은 적용되나, 공간은 차지 안 함 */
      margin: 5px; /* 수평 마진만 적용 */
    }
```

```
    .inline-block-example {
        display: inline-block; /* 같은 줄에 배치, 너비/높이 적용 가능 */
        background-color: lightcoral;
        margin-right: 10px;
    }

    .none-example {
        display: none; /* 완전히 숨김, 공간 차지 안 함 */
    }

    /* Position 속성 */
    .position-container {
        position: relative; /* absolute 자식의 기준점 */
        width: 400px;
        height: 300px;
        border: 2px solid blue;
        background-color: #e0e0ff;
        margin: 50px auto; /* 중앙 정렬 */
    }

    .static-box {
        background-color: yellow;
    }

    .relative-box {
        position: relative;
        top: 20px;
        left: 20px;
        background-color: orange;
        z-index: 2; /* 다른 요소 위에 표시 (z-index는 position이 static이 아닌
요소에만 적용) */
    }

    .absolute-box {
        position: absolute;
        top: 50px;
        right: 50px;
        background-color: purple;
        color: white;
        z-index: 3; /* 가장 위에 표시 */
    }

    .fixed-box {
        position: fixed;
```

```
            bottom: 20px;
            right: 20px;
            width: 150px;
            height: 70px;
            background-color: darkgreen;
            color: white;
            text-align: center;
            line-height: 70px;
            font-size: 1.2em;
            z-index: 100; /* 항상 화면에 고정, 맨 위에 표시 */
        }
    </style>
</head>
<body>
    <h1>Display 속성</h1>
    <div class="display-container">
        <div class="box block-example">블록 요소</div>
        <p>이것은 일반 단락입니다.</p>
        <span class="box inline-example">인라인 요소1</span>
        <span class="box inline-example">인라인 요소2</span>
        <div class="box inline-block-example">인라인-블록 요소1</div>
        <div class="box inline-block-example">인라인-블록 요소2</div>
        <div class="box none-example">숨겨진 요소</div>
        <p>이 단락은 숨겨진 요소 뒤에 바로 나타납니다. (공간이 없으므로)</p>
    </div>

    <h1>Position 및 Z-index 속성</h1>
    <p>아래 컨테이너 내에서 각 박스의 위치 변화를 확인하세요.</p>
    <div class="position-container">
        <div class="box static-box">Static Box (기준)</div>
        <div class="box relative-box">Relative Box</div>
        <div class="box absolute-box">Absolute Box</div>
        <div class="box" style="background-color: gray; position: absolute; top:
100px; left: 10px; z-index: 1;">
            Another Absolute Box (Z-index 1)
        </div>
    </div>

    <div class="fixed-box">고정된 박스</div>

    <p style="margin-top: 50px;">페이지를 스크롤하여 고정된 박스의 동작을
확인하세요. 이 텍스트는 페이지 하단에 도달할 때까지 스크롤 됩니다.</p>
    <p>...</p>
    <p>...</p>
    <p>...</p>
```

```
    <p>페이지를 아래로 스크롤하세요.</p>
    <p>...</p>
    <p>...</p>
    <p>...</p>
    <p>페이지의 맨 아래.</p>
</body>
</html>
```

2. 브라우저에서 확인**:**
    - Live Server로 display-position.html 파일을 열어 block, inline, inline-block 요소들이 어떻게 다르게 배치되는지 관찰합니다. 특히 inline 요소의 너비/높이 제한을 확인합니다.
    - position-container 내부의 static, relative, absolute 박스의 위치 변화를 유심히 살펴봅니다.
    - 페이지를 스크롤하면서 fixed-box가 뷰포트에 고정되어 있는 것을 확인합니다.
    - 개발자 도구 (F12)를 열어 각 요소의 "Computed" 탭에서 display와 position 속성 값을 확인하고, 박스 모델이 어떻게 변하는지 살펴봅니다.

8시간**:** 반응형 디자인 소개 **(Flexbox** 기본 및 미디어 쿼리**)**

이 시간은 반응형 웹 디자인(다양한 화면 크기에 맞게 레이아웃 조정)을 소개합니다. 기본적인 Flexbox 컨테이너 속성(display: flex, flex-direction, justify-content, align-items)과 조건부 스타일링을 위한 기본적인 미디어 쿼리(@media screen and (max-width:...))를 학습합니다. 모바일 우선 접근 방식의 개념(간략한 언급)과 유연하고 적응 가능한 레이아웃을 만드는 방법이 핵심 개념입니다. Flexbox와 기본적인 미디어 쿼리를 사용하여 작은 화면에서 수직으로 쌓이거나 단일 열로 축소되는 간단한 반응형 탐색 모음 또는 두 열 레이아웃을 구축하는 실습을 진행합니다.

핵심 개념**:**

- 반응형 웹 디자인이란?
- **Flexbox** (Flexible Box Layout):
    - 컨테이너 속성: display: flex, flex-direction, justify-content, align-items, flex-wrap
    - 자식 항목(flex-item) 속성 (간략히)
- 미디어 쿼리 (@media): 화면 크기에 따른 조건부 스타일링
    - max-width, min-width
- 모바일 우선 (Mobile-First) 디자인 개념 (간략히)
- 뷰포트 메타 태그 (<meta name="viewport">)

실습 예제**:**

1. 새로운 **HTML** 파일 생성**:**

- responsive-design.html 파일을 생성하고 다음 코드를 작성합니다.

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>반응형 디자인 예제 (Flexbox & 미디어 쿼리)</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 20px;
      background-color: #f4f4f4;
    }

    header {
      background-color: #333;
      color: white;
      padding: 1rem;
      text-align: center;
      margin-bottom: 20px;
    }

    .flex-container {
      display: flex; /* Flexbox 컨테이너 활성화 */
      flex-direction: row; /* 기본값: 가로 정렬 */
      justify-content: space-around; /* 항목들 사이에 공간 배분 */
      align-items: flex-start; /* 세로 시작 지점 정렬 */
      border: 2px solid #0056b3;
      padding: 15px;
      background-color: white;
      flex-wrap: wrap; /* 항목이 넘치면 다음 줄로 */
      gap: 10px; /* flex 항목 사이 간격 */
    }

    .flex-item {
      background-color: lightgray;
      border: 1px solid #777;
      padding: 20px;
      margin: 5px;
      min-width: 200px; /* 최소 너비 */
      max-width: 30%; /* 최대 너비 (더 많은 항목을 위해) */
      box-sizing: border-box; /* 패딩과 보더가 너비에 포함되도록 */
      text-align: center;
```

```
    flex-grow: 1; /* 남은 공간을 채우도록 성장 */
}

/* 간단한 반응형 네비게이션 예시 */
.navbar {
    display: flex;
    background-color: #444;
    padding: 10px;
    justify-content: space-around;
    flex-wrap: wrap;
}

.navbar a {
    color: white;
    text-decoration: none;
    padding: 10px 15px;
    margin: 5px;
    border-radius: 5px;
    transition: background-color 0.3s ease;
}

.navbar a:hover {
    background-color: #555;
}

/* 미디어 쿼리: 화면 너비가 768px 이하일 때 */
@media screen and (max-width: 768px) {
    .flex-container {
        flex-direction: column; /* 세로 정렬로 변경 */
        align-items: center; /* 세로 정렬 시 가운데 정렬 */
    }

    .flex-item {
        width: 90%; /* 작은 화면에서 너비를 더 넓게 */
        max-width: none; /* 최대 너비 제한 해제 */
    }

    .navbar {
        flex-direction: column; /* 네비게이션 항목 세로 정렬 */
        align-items: center;
    }

    .navbar a {
        width: 80%; /* 링크 너비 확장 */
        text-align: center;
```

```
        }
    }

    /* 미디어 쿼리: 화면 너비가 480px 이하일 때 (더 작은 화면) */
    @media screen and (max-width: 480px) {
        body {
            padding: 10px;
        }
        .flex-item {
            font-size: 0.9em;
        }
    }
</style>
</head>
<body>
<header>
    <h1>반응형 웹 디자인</h1>
</header>

<nav class="navbar">
    <a href="#">홈</a>
    <a href="#">서비스</a>
    <a href="#">제품</a>
    <a href="#">문의</a>
</nav>

<h2>Flexbox 레이아웃 예제</h2>
<p>브라우저 창의 너비를 조절하여 아래 Flexbox 컨테이너의 레이아웃 변화를 확인하세요.</p>
<div class="flex-container">
    <div class="flex-item">항목 1 - 긴 내용이 있을 수 있습니다.</div>
    <div class="flex-item">항목 2</div>
    <div class="flex-item">항목 3</div>
    <div class="flex-item">항목 4</div>
    <div class="flex-item">항목 5</div>
</div>

<p style="margin-top: 30px;">위 네비게이션 바도 반응형으로 동작합니다.</p>
</body>
</html>
```

2. 브라우저에서 확인:
    - Live Server로 responsive-design.html 파일을 열어봅니다.
    - 가장 중요한 실습: 브라우저 창의 너비를 줄이거나 늘려보세요. 768px 이하로

줄어들 때 Flexbox 항목들이 가로에서 세로로 변경되고, 내비게이션 바의 링크들도 세로로 쌓이는 것을 확인합니다.
    - 개발자 도구 (F12)를 열어 "Responsive Design Mode" (또는 "Device Toolbar")를 활성화하고 다양한 디바이스 크기에서 페이지가 어떻게 보이는지 시뮬레이션해 봅니다.

모듈 **3: JavaScript** 기초 **(4**시간**)**

9시간**: JavaScript** 소개 및 기본 구문

이 시간은 웹 페이지에 상호작용성을 추가하는 클라이언트 측 스크립팅 언어인 JavaScript의 역할을 다룹니다. HTML에 JS를 포함하는 방법(`

```
  </script>
</body>
</html>
```

2. 브라우저에서 확인**:**
    - Live Server로 js-basics.html 파일을 엽니다.
    - **개발자 도구 (F12 또는 Ctrl+Shift+I)**를 열고 "Console" 탭을 클릭합니다. HTML 파일에 작성된 JavaScript 코드가 출력하는 메시지들을 확인합니다.
    - 콘솔에 직접 JavaScript 코드를 입력하고 실행해봅니다.
        - let greeting = "Hello"; 입력 후 Enter
        - greeting 입력 후 Enter (변수 값 확인)
        - typeof greeting 입력 후 Enter (변수 타입 확인)
        - 10 + 20 입력 후 Enter
        - const myConstant = 100; 입력 후 Enter
        - myConstant = 200; 입력 후 Enter (에러 메시지 확인)

10시간**:** 연산자**,** 조건문 및 반복문

이 시간은 프로그래밍의 제어 흐름을 다룹니다. 일반적인 JavaScript 연산자(산술, 비교, 논리), 의사 결정을 위한 조건문(if, else if, else), 그리고 반복 작업을 위한 기본적인 반복 구조(for 루프, while 루프)를 학습합니다. 프로그램이 조건에 따라 결정을 내리고 코드 블록을 효율적으로 반복할 수 있도록 하는 것이 핵심 개념입니다. 연산자를 사용하여 간단한 계산기 로직을 구축하고, 조건문을 활용하는 기본적인 "숫자 맞추기" 게임을 만들며, 반복문을 사용하여 미리 정의된 항목 세트를 반복하는 실습을 진행합니다.

핵심 개념**:**

- **JavaScript** 연산자:

- 산술 연산자 (+, -, *, /, %, ** (거듭제곱))
- 할당 연산자 (=, +=, -=, *=, /=)
- 비교 연산자 (==, === (일치 연산자), !=, !==, >, <, >=, <=)
- 논리 연산자 (&& (AND), || (OR), ! (NOT))

- 조건문:
  - if 문
  - if...else 문
  - if...else if...else 문
  - switch 문 (간략히 언급)
- 반복문:
  - for 루프
  - while 루프
  - do...while 루프 (간략히 언급)

실습 예제**:**

1. 새로운 **HTML** 파일 생성**:**
   - operators-conditionals-loops.html 파일을 생성하고 다음 코드를 작성합니다.

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>JavaScript 연산자, 조건문, 반복문 예제</title>
</head>
<body>
  <h1>JavaScript 연산자, 조건문, 반복문</h1>
  <p>개발자 도구(F12)를 열고 콘솔 탭을 확인하세요.</p>

  <script>
    // 1. 연산자 예제
    console.log("--- 연산자 예제 ---");
    let numA = 20;
    let numB = 7;

    // 산술 연산자
    console.log("덧셈:", numA + numB);      // 27
    console.log("뺄셈:", numA - numB);      // 13
    console.log("곱셈:", numA * numB);      // 140
    console.log("나눗셈:", numA / numB);    // 약 2.857
    console.log("나머지:", numA % numB);    // 6
    console.log("거듭제곱 (2 ** 3):", 2 ** 3); // 8
```

```
// 할당 연산자
let x = 10;
x += 5; // x = x + 5; (x는 15)
console.log("x (+= 5):", x);
x *= 2; // x = x * 2; (x는 30)
console.log("x (*= 2):", x);

// 비교 연산자
console.log("10 == '10' (값만 비교):", 10 == '10');   // true
console.log("10 === '10' (값과 타입 비교):", 10 === '10'); // false (매우 중요!)
console.log("5 != '5' (값만 다름):", 5 != '5');     // false
console.log("5 !== '5' (값 또는 타입 다름):", 5 !== '5'); // true
console.log("numA > numB:", numA > numB);         // true

// 논리 연산자
let isAdult = true;
let hasLicense = false;
console.log("성인이면서 면허가 있는가?", isAdult && hasLicense); // false
console.log("성인이거나 면허가 있는가?", isAdult || hasLicense); // true
console.log("성인이 아닌가?", !isAdult);                    // false

// 2. 조건문 예제 (if-else if-else)
console.log("\n--- 조건문 예제 ---");
let userScore = 75;
let grade;

if (userScore >= 90) {
    grade = "A";
} else if (userScore >= 80) {
    grade = "B";
} else if (userScore >= 70) {
    grade = "C";
} else if (userScore >= 60) {
    grade = "D";
} else {
    grade = "F";
}
console.log("사용자 점수:", userScore, ", 학점:", grade);

// 다른 예시: 간단한 숫자 맞추기 게임 (콘솔에서 직접 입력)
// const targetNumber = Math.floor(Math.random() * 10) + 1; // 1~10 랜덤 숫자
// let guess = parseInt(prompt("1부터 10 사이의 숫자를 맞춰보세요!"));
// if (guess === targetNumber) {
//     console.log("정답입니다!");
// } else if (guess < targetNumber) {
```

```
        //     console.log("너무 작아요!");
        // } else {
        //     console.log("너무 커요!");
        // }

        // 3. 반복문 예제
        console.log("\n--- 반복문 예제 ---");

        // for 루프: 특정 횟수만큼 반복할 때 유용
        console.log("For 루프 (0부터 4까지 출력):");
        for (let i = 0; i < 5; i++) {
            console.log("반복 횟수:", i);
        }

        // 배열 요소 반복 (다음 시간 예고편)
        const fruits = ["사과", "바나나", "오렌지"];
        console.log("For 루프 (과일 목록 출력):");
        for (let i = 0; i < fruits.length; i++) {
            console.log(fruits[i] + "를(을) 좋아합니다.");
        }

        // while 루프: 조건이 참인 동안 반복
        console.log("While 루프 (카운트 다운):");
        let count = 3;
        while (count > 0) {
            console.log("카운트:", count);
            count--; // count = count - 1;
        }
        console.log("발사!");

        // 무한 루프 주의! (조건문이 항상 참일 경우)
        // while (true) { console.log("무한 루프!"); }
    </script>
</body>
</html>
```

2. 브라우저에서 확인:
    - Live Server로 operators-conditionals-loops.html 파일을 엽니다.
    - **개발자 도구 (F12)**의 "Console" 탭을 열어 스크립트가 출력하는 다양한 연산 결과, 조건문 결과, 반복문 실행 결과를 확인합니다.
    - 특히 ==와 ===의 차이점을 명확히 이해하도록 설명하고, 직접 콘솔에서 여러 값으로 실험하게 합니다.
    - 간단한 prompt() 함수를 사용하여 사용자 입력을 받아 조건문 결과를

테스트하는 것도 좋은 실습이 될 수 있습니다. (예시 코드 주석 참고)

11시간**:** 함수 및 배열

이 시간은 코드 재사용성을 위한 함수(선언, 호출, 매개변수, 반환 값)와 데이터 컬렉션을 구성하기 위한 배열(생성, 요소 접근, push, pop, length와 같은 일반적인 메서드)을 다룹니다. 함수로 코드를 모듈화하는 이점, 함수로 데이터를 전달하고 반환하는 방법, 그리고 정렬된 데이터 목록을 효율적으로 관리하는 것이 핵심 개념입니다. 재사용 가능한 작업을 위한 함수(예: 합계를 계산하는 함수)를 만들고, 배열 내의 데이터를 조작하는 실습(예: 간단한 할 일 목록 백엔드 로직에서 항목 추가/제거)을 진행합니다.

핵심 개념**:**

- 함수 **(Functions)**:
  - 함수 선언 (Function Declaration)
  - 함수 표현식 (Function Expression)
  - 매개변수 (Parameters)와 인자 (Arguments)
  - return 문을 사용한 값 반환
  - 코드 재사용성, 모듈화
- 배열 **(Arrays)**:
  - 배열 생성 (대괄호 [], new Array())
  - 인덱스를 사용한 요소 접근 (0부터 시작)
  - 배열의 길이 (.length 속성)
  - 주요 배열 메서드: push(), pop(), shift(), unshift(), splice() (간략히), forEach() (간략히)

실습 예제**:**

1. 새로운 **HTML** 파일 생성**:**
   - functions-arrays.html 파일을 생성하고 다음 코드를 작성합니다.

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>JavaScript 함수 및 배열 예제</title>
</head>
<body>
  <h1>JavaScript 함수 및 배열</h1>
  <p>개발자 도구(F12)를 열고 콘솔 탭을 확인하세요.</p>

  <script>
    // 1. 함수 예제
    console.log("--- 함수 예제 ---");
```

```
// 함수 선언: 가장 일반적인 함수 정의 방식
function greet(name) { // 'name'은 매개변수
    return "안녕하세요, " + name + "님!"; // 값 반환
}

let greetingMessage = greet("김철수"); // 함수 호출, '김철수'는 인자
console.log(greetingMessage); // "안녕하세요, 김철수님!"

console.log(greet("영희")); // "안녕하세요, 영희님!"

// 두 숫자의 합계를 계산하는 함수
function addNumbers(num1, num2) {
    let sum = num1 + num2;
    return sum;
}

let result1 = addNumbers(10, 5);
console.log("10 + 5 =", result1); // 15

let result2 = addNumbers(100, 20);
console.log("100 + 20 =", result2); // 120

// 반환 값이 없는 함수 (콘솔에 출력만 하는 함수)
function showMessage(msg) {
    console.log("메시지: " + msg);
}
showMessage("함수는 재사용 가능한 코드 블록입니다.");

// 2. 배열 예제
console.log("\n--- 배열 예제 ---");

// 배열 생성
const fruits = ["사과", "바나나", "오렌지", "딸기"]; // 문자열 배열
const numbers = [1, 5, 8, 12, 3]; // 숫자 배열
const mixedArray = ["Hello", 123, true, null]; // 혼합 데이터 타입 배열

// 배열 요소 접근 (인덱스는 0부터 시작)
console.log("첫 번째 과일:", fruits[0]); // "사과"
console.log("세 번째 과일:", fruits[2]); // "오렌지"

// 배열의 길이
console.log("과일 배열의 길이:", fruits.length); // 4
console.log("숫자 배열의 길이:", numbers.length); // 5
```

```
    // 배열 메서드: 요소 추가/제거
    // push(): 배열 끝에 요소 추가
    fruits.push("포도");
    console.log("포도 추가 후:", fruits); // ["사과", "바나나", "오렌지", "딸기",
"포도"]

    // pop(): 배열 끝 요소 제거 (제거된 요소 반환)
    let removedFruit = fruits.pop();
    console.log("제거된 과일:", removedFruit); // "포도"
    console.log("포도 제거 후:", fruits); // ["사과", "바나나", "오렌지", "딸기"]

    // unshift(): 배열 시작에 요소 추가
    fruits.unshift("망고");
    console.log("망고 추가 후:", fruits); // ["망고", "사과", "바나나", "오렌지",
"딸기"]

    // shift(): 배열 시작 요소 제거 (제거된 요소 반환)
    let firstRemovedFruit = fruits.shift();
    console.log("제거된 첫 번째 과일:", firstRemovedFruit); // "망고"
    console.log("망고 제거 후:", fruits); // ["사과", "바나나", "오렌지", "딸기"]

    // forEach() 루프 (배열의 모든 요소 반복)
    console.log("과일 목록 (forEach):");
    fruits.forEach(function(fruit, index) {
      console.log(`${index + 1}. ${fruit}`);
    });

    // 실습: 간단한 할 일 목록 로직 (배열 이용)
    let todos = []; // 빈 할 일 목록 배열

    function addTodo(task) {
      todos.push(task);
      console.log(`'${task}' 추가됨. 현재 할 일 목록:`, todos);
    }

    function removeTodo(task) {
      const index = todos.indexOf(task); // 배열에서 task의 인덱스 찾기
      if (index > -1) { // 해당 task가 배열에 존재하면
        todos.splice(index, 1); // 해당 인덱스에서 1개 제거
        console.log(`'${task}' 제거됨. 현재 할 일 목록:`, todos);
      } else {
        console.log(`'${task}'는 할 일 목록에 없습니다.`);
      }
    }
```

```
        addTodo("아침 식사");
        addTodo("HTML 공부");
        addTodo("운동하기");
        removeTodo("HTML 공부");
        removeTodo("청소하기"); // 없는 항목 제거 시도
    </script>
</body>
</html>
```

2. 브라우저에서 확인**:**
    - Live Server로 functions-arrays.html 파일을 엽니다.
    - **개발자 도구 (F12)**의 "Console" 탭을 열어 함수 호출 결과와 배열 조작 결과를 확인합니다.
    - 콘솔에서 직접 함수를 호출하고 배열 메서드를 사용해 봅니다.
        - greet("새로운 이름");
        - addNumbers(50, 25);
        - todos (현재 todos 배열 상태 확인)
        - addTodo("새로운 작업");
        - removeTodo("운동하기");

12시간**: DOM** 조작 및 이벤트

이 시간은 JavaScript가 HTML과 상호작용하는 인터페이스인 문서 객체 모델(DOM)을 소개합니다. 요소 선택 메서드(예: document.querySelector, document.getElementById), 요소 수정(텍스트 콘텐츠, 속성, CSS 스타일 변경), 그리고 기본적인 이벤트 처리(addEventListener, click, mouseover, submit과 같은 일반적인 이벤트)를 학습합니다. JavaScript가 웹 페이지의 HTML 구조와 CSS 스타일링을 어떻게 상호작용하고 수정하는지, 그리고 웹 페이지가 사용자 상호작용에 어떻게 반응하도록 하는지 이해하는 것이 핵심 개념입니다. 버튼 클릭 시 텍스트나 이미지 소스를 변경하고, CSS 클래스를 토글하여 요소 모양을 변경하며, 기본적인 폼 유효성 검사 피드백(예: 입력이 비어 있으면 오류 메시지 표시)을 구현하는 실습을 진행합니다.

핵심 개념**:**

- 문서 객체 모델 **(DOM)** 이해: HTML 문서를 객체로 표현
- 요소 선택 **(Selecting Elements)**:
    - document.getElementById()
    - document.querySelector()
    - document.querySelectorAll()
- **DOM** 조작 **(Manipulating Elements)**:
    - 텍스트 콘텐츠 변경: textContent, innerHTML
    - 속성 변경: element.attribute = value, element.setAttribute(),

element.removeAttribute()
    - CSS 스타일 변경: element.style.property = value
    - CSS 클래스 조작: element.classList.add(), remove(), toggle()
    - 새로운 요소 생성 및 추가: document.createElement(), appendChild()
- 이벤트 처리 **(Event Handling)**:
    - 이벤트 리스너 등록: element.addEventListener(eventName, function)
    - 일반적인 이벤트: click, mouseover, mouseout, submit, input, keydown 등
    - 이벤트 객체 (event)

실습 예제**:**

1. 새로운 **HTML** 파일 생성**:**
    - dom-events.html 파일을 생성하고 다음 코드를 작성합니다.

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>JavaScript DOM 및 이벤트 예제</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
        }
        #myDiv {
            width: 250px;
            height: 120px;
            background-color: lightblue;
            border: 2px solid navy;
            text-align: center;
            line-height: 120px; /* 세로 중앙 정렬 */
            margin-top: 20px;
            cursor: pointer;
            transition: background-color 0.3s ease; /* 부드러운 전환 효과 */
        }
        .highlight {
            background-color: yellow;
            border-color: orange;
            color: #333;
            font-weight: bold;
        }
        #imageContainer img {
            width: 200px;
            height: auto;
```

```
            margin-top: 20px;
            border: 1px solid #ccc;
        }
        .error-message {
            color: red;
            font-size: 0.9em;
            margin-top: 5px;
        }
        input[type="text"], input[type="email"] {
            padding: 8px;
            margin-bottom: 10px;
            border: 1px solid #ccc;
            border-radius: 4px;
            width: 200px;
        }
        button {
            padding: 10px 15px;
            background-color: #007bff;
            color: white;
            border: none;
            border-radius: 5px;
            cursor: pointer;
            margin-right: 10px;
        }
        button:hover {
            background-color: #0056b3;
        }
    </style>
</head>
<body>
    <h1 id="mainTitle">환영합니다! JavaScript DOM 조작</h1>

    <button id="changeTextBtn">제목 텍스트 변경</button>
    <button id="toggleStyleBtn">스타일 토글</button>
    <button id="addImageBtn">이미지 추가</button>

    <div id="myDiv">클릭하세요!</div>

    <div id="imageContainer">
        <!-- 이미지가 여기에 동적으로 추가될 것입니다. -->
    </div>

    <h2>폼 유효성 검사 예제</h2>
    <form id="myForm">
        <label for="username">사용자 이름:</label><br>
```

```
    <input type="text" id="username" name="username"><br>
    <div id="usernameError" class="error-message"></div><br>

    <label for="useremail">이메일:</label><br>
    <input type="email" id="useremail" name="useremail"><br>
    <div id="useremailError" class="error-message"></div><br>

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

  <script>
    // 1. 요소 선택
    const mainTitle = document.querySelector("#mainTitle"); // ID로 요소 선택
    const changeTextButton = document.getElementById("changeTextBtn"); //
ID로 요소 선택 (다른 방법)
    const toggleStyleButton = document.querySelector("#toggleStyleBtn");
    const myDiv = document.querySelector("#myDiv");
    const addImageButton = document.getElementById("addImageBtn");
    const imageContainer = document.getElementById("imageContainer");
    const myForm = document.getElementById("myForm");
    const usernameInput = document.getElementById("username");
    const useremailInput = document.getElementById("useremail");
    const usernameError = document.getElementById("usernameError");
    const useremailError = document.getElementById("useremailError");

    // 2. DOM 조작 (텍스트 콘텐츠 변경)
    changeTextButton.addEventListener("click", function() {
      mainTitle.textContent = "JavaScript로 텍스트가 변경되었습니다!"; // 텍스트
콘텐츠 변경
      // 또는 mainTitle.innerHTML = "<em>JavaScript</em>로 텍스트가
변경되었습니다!"; (HTML 태그 포함)
    });

    // 3. DOM 조작 (스타일 변경 - 클래스 토글)
    toggleStyleButton.addEventListener("click", function() {
      myDiv.classList.toggle("highlight"); // CSS 클래스 추가/제거
      // myDiv.style.backgroundColor = "pink"; // 직접 스타일 변경
    });

    // 4. 새로운 요소 생성 및 추가 (이미지 추가)
    addImageButton.addEventListener("click", function() {
      const newImage = document.createElement("img"); // <img> 요소 생성
      newImage.src =
"https://via.placeholder.com/200x150/FF5733/FFFFFF?text=New+Image"; //
이미지 소스 설정
```

```
    newImage.alt = "새롭게 추가된 이미지"; // 대체 텍스트 설정
    imageContainer.appendChild(newImage); // imageContainer에 이미지 추가
  });

  // 5. 이벤트 처리 (클릭 이벤트)
  myDiv.addEventListener("click", function() {
    alert("div가 클릭되었습니다! 배경색이 변경됩니다."); // 경고창 표시
    myDiv.style.backgroundColor = "lightgreen"; // 직접 스타일 변경
  });

  myDiv.addEventListener("mouseover", function() {
    myDiv.textContent = "마우스를 올렸습니다!";
  });

  myDiv.addEventListener("mouseout", function() {
    myDiv.textContent = "클릭하세요!";
    myDiv.style.backgroundColor = "lightblue";
  });

  // 6. 폼 유효성 검사 및 제출 이벤트
  myForm.addEventListener("submit", function(event) {
    event.preventDefault(); // 폼 기본 제출 동작 방지 (페이지 새로고침 방지)

    let isValid = true;

    // 사용자 이름 유효성 검사
    if (usernameInput.value.trim() === "") { // .trim()은 앞뒤 공백 제거
      usernameError.textContent = "사용자 이름을 입력하세요.";
      isValid = false;
    } else {
      usernameError.textContent = ""; // 오류 메시지 제거
    }

    // 이메일 유효성 검사
    if (useremailInput.value.trim() === "" || !useremailInput.value.includes('@'))
{
      useremailError.textContent = "유효한 이메일 주소를 입력하세요.";
      isValid = false;
    } else {
      useremailError.textContent = "";
    }

    if (isValid) {
      alert("폼이 성공적으로 제출되었습니다!");
      console.log("제출된 데이터:", {
```

```
            username: usernameInput.value,
            useremail: useremailInput.value
        });
        // 실제 서버로 데이터 전송 로직은 여기에 들어감 (AJAX 등)
        myForm.reset(); // 폼 필드 초기화
    } else {
        console.log("폼 제출 실패: 유효성 검사 오류");
    }
});
</script>
</body>
</html>
```

2. 브라우저에서 확인**:**
    - Live Server로 dom-events.html 파일을 엽니다.
    - "제목 텍스트 변경" 버튼을 클릭하여 제목이 바뀌는 것을 확인합니다.
    - "스타일 토글" 버튼을 클릭하여 myDiv의 배경색과 테두리가 변경되는 것을 확인합니다.
    - myDiv 위로 마우스를 올렸다 내렸다 해보고, 클릭하여 경고창이 뜨고 배경색이 변하는 것을 확인합니다.
    - "이미지 추가" 버튼을 클릭하여 새로운 이미지가 페이지에 추가되는 것을 확인합니다.
    - 폼에 아무것도 입력하지 않고 "제출" 버튼을 클릭하여 오류 메시지가 뜨는 것을 확인합니다. 유효한 값을 입력한 후 제출하여 성공 메시지를 확인합니다.
    - 개발자 도구의 "Elements" 탭에서 DOM 변화를 실시간으로 확인하고, "Console" 탭에서 오류 메시지나 console.log 출력 결과를 확인합니다.

모듈 **4:** 통합 프로젝트 및 다음 단계 **(4**시간**)**

13-14시간**:** 미니 프로젝트 구축 **(**파트 **1)**

이 시간은 프로젝트 계획(큰 작업을 관리 가능한 HTML, CSS, JS 구성 요소로 분해), 선택한 프로젝트를 위한 HTML 구조화, 그리고 기본적인 반응형 디자인 원리(Flexbox, 간단한 미디어 쿼리)를 포함한 CSS 스타일링 적용을 다룹니다. 학습한 모든 HTML 및 CSS 개념의 통합, 완전한 웹 페이지를 구축하기 위한 체계적인 워크플로우 개발이 핵심 개념입니다. 간단한 다중 섹션 웹 페이지를 안내에 따라 구축하는 실습을 진행합니다. 예를 들어, 개인 포트폴리오 페이지(소개, 기술, 연락처 섹션), 기본적인 제품 랜딩 페이지, 또는 간단한 대화형 도구의 사용자 인터페이스(아직 전체 JS 로직이 없는 기본적인 계산기 UI) 등이 될 수 있습니다.

핵심 개념**:**

- 프로젝트 기반 학습의 중요성

- 프로젝트 계획: HTML, CSS, JS 구성 요소로 분해
- **HTML** 시맨틱 구조화 재확인
- **CSS**를 이용한 레이아웃 및 스타일링 통합 (**Flexbox**, 미디어 쿼리 포함)
- 체계적인 개발 워크플로우 (파일 구조, 주석 등)

실습 예제**:** 간단한 개인 포트폴리오 웹사이트 구축 **(HTML & CSS)**

목표**:** 소개, 기술, 프로젝트, 연락처 섹션으로 구성된 단일 페이지 포트폴리오 웹사이트의 HTML 구조를 만들고, 기본적인 CSS를 적용하여 반응형 레이아웃을 구현합니다.

파일 구조**:**

```
my-portfolio/
├── index.html
└── style.css
```

1. index.html 작성**:**

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>내 포트폴리오 - [당신의 이름]</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <header>
        <nav class="navbar">
            <div class="logo">[당신의 이름] 포트폴리오</div>
            <ul class="nav-links">
                <li><a href="#home">홈</a></li>
                <li><a href="#about">소개</a></li>
                <li><a href="#skills">기술</a></li>
                <li><a href="#projects">프로젝트</a></li>
                <li><a href="#contact">문의</a></li>
            </ul>
        </nav>
    </header>
```

```
  <main>
    <section id="home" class="hero-section">
      <div class="hero-content">
        <h1>안녕하세요, 저는 [당신의 이름]입니다!</h1>
        <p>프런트엔드 웹 개발을 배우고 있는 주니어 개발자입니다.</p>
        <a href="#contact" class="btn">연락하기</a>
      </div>
    </section>

    <section id="about" class="about-section common-section">
      <h2>소개</h2>
      <div class="about-content">
        <img src="https://via.placeholder.com/150" alt="프로필 사진"
class="profile-pic">
        <p>
          저는 [당신의 이름]이며, 웹 개발의 매력에 빠져 끊임없이 배우고 성장하고
있습니다.
          특히 사용자 경험을 개선하고 아름다운 인터페이스를 만드는 데 열정을
가지고 있습니다.
          이 과정에서 HTML, CSS, JavaScript의 기본을 탄탄히 다졌습니다.
        </p>
        <p>
          새로운 기술을 익히고 실제 프로젝트에 적용하는 것을 즐깁니다.
          지속적인 학습을 통해 프런트엔드 개발자로서의 역량을 강화하고 있습니다.
        </p>
      </div>
    </section>

    <section id="skills" class="skills-section common-section">
      <h2>기술 스택</h2>
      <div class="skill-grid">
        <div class="skill-item">
          <h3>HTML5</h3>
          <p>웹 콘텐츠의 구조를 정의하는 데 능숙합니다.</p>
        </div>
        <div class="skill-item">
          <h3>CSS3</h3>
          <p>Flexbox와 미디어 쿼리를 사용한 반응형 디자인을 이해하고
있습니다.</p>
```

```
            </div>
            <div class="skill-item">
                <h3>JavaScript</h3>
                <p>DOM 조작 및 이벤트 처리를 통한 동적 웹 페이지 구현이 가능합니다.</p>
            </div>
            <div class="skill-item">
                <h3>Git & GitHub (기초)</h3>
                <p>버전 관리 시스템의 기본 개념을 이해하고 코드 공유에 활용합니다.</p>
            </div>
        </div>
    </section>

    <section id="projects" class="projects-section common-section">
        <h2>프로젝트</h2>
        <div class="project-grid">
            <div class="project-item">
                <h3>프로젝트 1: 랜딩 페이지</h3>
                <p>간단한 기업 랜딩 페이지. HTML 시맨틱 태그와 CSS Flexbox를 활용하여
반응형으로 제작.</p>
                <a href="#" target="_blank" class="project-link">자세히 보기</a>
            </div>
            <div class="project-item">
                <h3>프로젝트 2: 할 일 목록 앱 (UI만)</h3>
                <p>JavaScript 로직은 없지만, 사용자가 할 일을 추가/삭제할 수 있는 UI 구현
연습.</p>
                <a href="#" target="_blank" class="project-link">자세히 보기</a>
            </div>
        </div>
    </section>

    <section id="contact" class="contact-section common-section">
        <h2>문의하기</h2>
        <p>궁금한 점이 있다면 언제든지 연락 주세요!</p>
        <form class="contact-form">
            <label for="name-contact">이름:</label>
            <input type="text" id="name-contact" name="name" required>

            <label for="email-contact">이메일:</label>
            <input type="email" id="email-contact" name="email" required>
```

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

            <button type="submit">메시지 보내기</button>
          </form>
        </section>
    </main>

    <footer>
        <p>© 2025 [당신의 이름]. 모든 권리 보유.</p>
    </footer>
</body>
</html>
```

2. style.css 작성**:**

```
/* style.css */

/* 기본 초기화 */
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    line-height: 1.6;
    color: #333;
    background-color: #f4f4f4;
}

/* 네비게이션 바 */
.navbar {
    display: flex;
    justify-content: space-between;
    align-items: center;
```

```
    background-color: #333;
    color: white;
    padding: 1rem 2rem;
    flex-wrap: wrap; /* 작은 화면에서 줄 바꿈 허용 */
}

.navbar .logo {
    font-size: 1.5rem;
    font-weight: bold;
    margin-right: 20px; /* 로고와 링크 사이 간격 */
}

.navbar .nav-links {
    list-style: none;
    display: flex;
    gap: 1.5rem; /* 링크 사이 간격 */
    flex-wrap: wrap;
}

.navbar .nav-links a {
    color: white;
    text-decoration: none;
    font-weight: bold;
    transition: color 0.3s ease;
}

.navbar .nav-links a:hover {
    color: #007bff;
}

/* 히어로 섹션 */
.hero-section {
    background: linear-gradient(rgba(0, 0, 0, 0.6), rgba(0, 0, 0, 0.6)),
url('https://via.placeholder.com/1500x500/007bff/FFFFFF?text=Hero+Background')
no-repeat center center/cover;
    color: white;
    text-align: center;
    padding: 6rem 2rem;
    display: flex;
```

```
    justify-content: center;
    align-items: center;
    min-height: 500px;
}

.hero-content h1 {
    font-size: 3.5rem;
    margin-bottom: 1rem;
}

.hero-content p {
    font-size: 1.5rem;
    margin-bottom: 2rem;
}

.btn {
    display: inline-block;
    background-color: #007bff;
    color: white;
    padding: 0.8rem 2rem;
    text-decoration: none;
    border-radius: 5px;
    transition: background-color 0.3s ease;
}

.btn:hover {
    background-color: #0056b3;
}

/* 공통 섹션 스타일 */
.common-section {
    padding: 4rem 2rem;
    margin: 0 auto;
    max-width: 1200px;
    background-color: white;
    border-bottom: 1px solid #eee;
}

.common-section:last-of-type {
```

```
    border-bottom: none;
}

.common-section h2 {
    text-align: center;
    font-size: 2.5rem;
    margin-bottom: 3rem;
    color: #333;
    position: relative;
}

.common-section h2::after {
    content: '';
    display: block;
    width: 60px;
    height: 3px;
    background-color: #007bff;
    margin: 10px auto 0;
}

/* 소개 섹션 */
.about-content {
    display: flex;
    flex-direction: column;
    align-items: center;
    text-align: center;
    gap: 20px;
}

.profile-pic {
    border-radius: 50%;
    width: 180px;
    height: 180px;
    object-fit: cover;
    border: 5px solid #007bff;
    margin-bottom: 20px;
}

.about-content p {
```

```
    font-size: 1.1rem;
    max-width: 800px;
}

/* 기술 섹션 */
.skill-grid, .project-grid {
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    gap: 25px;
}

.skill-item, .project-item {
    background-color: #f9f9f9;
    border: 1px solid #ddd;
    border-radius: 8px;
    padding: 25px;
    text-align: center;
    flex: 1 1 calc(33% - 40px); /* 3개씩, gap 고려 */
    min-width: 280px;
    box-shadow: 0 2px 5px rgba(0,0,0,0.1);
    transition: transform 0.3s ease;
}

.skill-item:hover, .project-item:hover {
    transform: translateY(-5px);
}

.skill-item h3, .project-item h3 {
    color: #007bff;
    margin-bottom: 10px;
    font-size: 1.4rem;
}

.skill-item p, .project-item p {
    font-size: 1rem;
    color: #555;
}
```

```
/* 프로젝트 섹션 */
.project-link {
    display: inline-block;
    margin-top: 15px;
    color: #007bff;
    text-decoration: none;
    font-weight: bold;
}

.project-link:hover {
    text-decoration: underline;
}

/* 문의 섹션 */
.contact-form {
    display: flex;
    flex-direction: column;
    max-width: 600px;
    margin: 0 auto;
    padding: 20px;
    border: 1px solid #eee;
    border-radius: 8px;
    background-color: #f9f9f9;
}

.contact-form label {
    margin-bottom: 8px;
    font-weight: bold;
    color: #555;
}

.contact-form input,
.contact-form textarea {
    padding: 12px;
    margin-bottom: 15px;
    border: 1px solid #ccc;
    border-radius: 4px;
    font-size: 1rem;
    width: 100%;
```

```
}

.contact-form input:focus,
.contact-form textarea:focus {
    outline: none;
    border-color: #007bff;
    box-shadow: 0 0 5px rgba(0, 123, 255, 0.2);
}

.contact-form button[type="submit"] {
    background-color: #28a745;
    color: white;
    padding: 12px 20px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    font-size: 1.1rem;
    transition: background-color 0.3s ease;
}

.contact-form button[type="submit"]:hover {
    background-color: #218838;
}


/* 푸터 */
footer {
    text-align: center;
    padding: 2rem;
    background-color: #333;
    color: white;
    font-size: 0.9rem;
}

/* 미디어 쿼리 (반응형 디자인) */
@media (max-width: 768px) {
    .navbar {
        flex-direction: column;
        align-items: flex-start;
```

```
        gap: 15px;
    }

    .navbar .nav-links {
        flex-direction: column;
        width: 100%;
        text-align: left;
        gap: 10px;
    }

    .hero-content h1 {
        font-size: 2.5rem;
    }

    .hero-content p {
        font-size: 1.2rem;
    }

    .common-section {
        padding: 2.5rem 1rem;
    }

    .skill-item, .project-item {
        flex: 1 1 100%; /* 한 줄에 하나씩 */
        max-width: none;
    }

    .profile-pic {
        width: 150px;
        height: 150px;
    }
}

@media (max-width: 480px) {
    .hero-content h1 {
        font-size: 2rem;
    }

    .hero-content p {
```

```
    font-size: 1rem;
  }

  .btn {
    padding: 0.6rem 1.5rem;
    font-size: 0.9rem;
  }

  .navbar .logo {
    font-size: 1.2rem;
  }
}
```

실습 진행**:**

1. index.html과 style.css 파일을 위 내용으로 생성합니다.
2. Live Server로 index.html 파일을 엽니다.
3. 브라우저 창 크기를 조절하면서 내비게이션 바, 히어로 섹션, 기술 섹션 등의 레이아웃이 어떻게 반응하는지 확인합니다.
4. CSS 파일에서 색상, 글꼴 크기, 패딩, 마진 등을 변경하여 디자인을 커스터마이징 해봅니다.
5. 이미지 URL(https://via.placeholder.com/)을 자신의 이미지로 변경하거나, 실제 프로젝트 이미지로 대체해 봅니다.
6. 폼 필드에 placeholder 속성을 추가하고 required 속성을 사용하여 간단한 HTML5 유효성 검사를 확인합니다.
7. main 태그 내에 있는 각 section에 id를 부여하고, 내비게이션 바의 a 태그 href 속성에 해당 id를 연결하여 페이지 내 이동을 구현해봅니다.

15시간**:** 미니 프로젝트 구축 **(**파트 **2)** 및 기본 배포

이 시간은 프로젝트를 위한 JavaScript 상호작용 구현(예: 기본적인 폼 유효성 검사, 사용자 입력에 따른 동적 콘텐츠 업데이트, 간단한 이미지 슬라이더 또는 탭 콘텐츠), 그리고 웹 호스팅(GitHub Pages를 사용한 정적 사이트 배포) 소개를 다룹니다. 프런트엔드 로직으로 프로젝트에 생명을 불어넣고, 프로젝트를 온라인에서 접근 가능하게 만드는 방법을 이해하는 것이 핵심 개념입니다. JavaScript 기능을 추가하여 프로젝트를 계속 구축한 다음, 완성된 정적 사이트를 GitHub Pages에 배포하여 공개적으로 접근 가능하게 만드는 실습을 진행합니다.

핵심 개념**:**

- **JavaScript**를 이용한 동적 상호작용 구현

- **DOM** 조작 및 이벤트 리스너 활용 심화
- 간단한 폼 유효성 검사 (JavaScript 활용)
- 정적 사이트 배포의 개념
- **GitHub Pages**를 이용한 웹사이트 배포 (**Git** 기초 포함)

실습 예제**:** 포트폴리오 웹사이트에 **JavaScript** 추가 및 **GitHub Pages** 배포

목표**:** 이전 시간의 포트폴리오 웹사이트에 JavaScript를 추가하여 간단한 동적 기능을 구현하고, 완성된 웹사이트를 GitHub Pages를 통해 온라인에 배포합니다.

파일 구조 **(**추가**):**

```
my-portfolio/
├── index.html
├── style.css
└── script.js  <-- 이 파일이 새로 추가됩니다.
```

1. script.js 작성 **(**새로운 파일**):**

```
// script.js

// 1. 네비게이션 스크롤 효과 (부드러운 스크롤)
document.querySelectorAll('.nav-links a').forEach(anchor => {
    anchor.addEventListener('click', function (e) {
        e.preventDefault(); // 기본 앵커 동작 방지

        const targetId = this.getAttribute('href'); // #home, #about 등
        const targetElement = document.querySelector(targetId);

        if (targetElement) {
            window.scrollTo({
                top: targetElement.offsetTop - 
document.querySelector('header').offsetHeight, // 헤더 높이만큼 빼서 정확한 위치로 이동
                behavior: 'smooth' // 부드러운 스크롤
            });
        }
    });
});
```

```
// 2. 문의하기 폼 유효성 검사 (JavaScript 버전)
const contactForm = document.querySelector('.contact-form');
const nameInput = document.getElementById('name-contact');
const emailInput = document.getElementById('email-contact');
const messageInput = document.getElementById('message');

contactForm.addEventListener('submit', function(event) {
    event.preventDefault(); // 기본 제출 동작 방지

    let isValid = true;

    // 이름 유효성 검사
    if (nameInput.value.trim() === '') {
        alert('이름을 입력해주세요.');
        isValid = false;
        nameInput.focus(); // 해당 필드로 포커스 이동
        return; // 추가 검사 중단
    }

    // 이메일 유효성 검사 (간단한 형식)
    if (emailInput.value.trim() === '' || !emailInput.value.includes('@') ||
!emailInput.value.includes('.')) {
        alert('유효한 이메일 주소를 입력해주세요.');
        isValid = false;
        emailInput.focus();
        return;
    }

    // 메시지 유효성 검사
    if (messageInput.value.trim() === '') {
        alert('메시지를 입력해주세요.');
        isValid = false;
        messageInput.focus();
        return;
    }

    if (isValid) {
        // 모든 유효성 검사를 통과했을 때
```

```
        alert('메시지가 성공적으로 전송되었습니다!');
        console.log('폼 제출 데이터:', {
            name: nameInput.value,
            email: emailInput.value,
            message: messageInput.value
        });
        contactForm.reset(); // 폼 필드 초기화
        // 실제 서버로 데이터를 전송하는 AJAX 요청은 나중에 학습할 내용입니다.
    }
});


// 3. 스크롤 시 헤더 배경색 변경 (선택 사항)
const header = document.querySelector('header');

window.addEventListener('scroll', function() {
    if (window.scrollY > 50) { // 50px 이상 스크롤되면
        header.style.backgroundColor = 'rgba(51, 51, 51, 0.9)'; // 투명도 있는 배경색
    } else {
        header.style.backgroundColor = '#333'; // 원래 배경색
    }
});
```

2. index.html 수정**:**

<head> 섹션의 <link rel="stylesheet" href="style.css"> 아래에 다음 줄을 추가하여 script.js 파일을 연결합니다. **</body>** 닫는 태그 직전에 넣는 것이 좋습니다**.** (HTML이 먼저 로드되고 나서 JS가 DOM 요소를 조작할 수 있도록)

```
    <!-- ... head 부분 ... -->
    <script src="script.js" defer></script>
</body>
</html>
```
```defer` 속성은 HTML 파싱이 끝난 후에 스크립트가 실행되도록 합니다.

실습 진행 (JavaScript 기능 확인):

1. Live Server로 `index.html` 파일을 엽니다.
2. 내비게이션 링크를 클릭하여 페이지가 부드럽게 스크롤되는지 확인합니다.
```

3. 문의하기 폼에서 빈 필드나 잘못된 이메일 형식으로 제출을 시도하여 JavaScript 기반의 유효성 검사가 작동하는지 확인합니다.
4. 스크롤을 내렸을 때 헤더의 배경색이 변하는지 확인합니다.
5. 개발자 도구 (F12)의 콘솔 탭에서 폼 제출 시 출력되는 메시지를 확인합니다.

GitHub Pages를 이용한 웹사이트 배포 (단계별 가이드):

1. **Git 설치:**
   * Git이 설치되어 있지 않다면, Git 공식 웹사이트에서 다운로드하여 설치합니다. (터미널에서 `git --version`으로 확인 가능)

2. **GitHub 계정 생성:**
   * GitHub에 계정이 없다면, GitHub 웹사이트에서 가입합니다.

3. **새 저장소 (Repository) 생성:**
   * GitHub에 로그인 후, 화면 오른쪽 상단의 '+' 버튼을 클릭하여 "New repository"를 선택합니다.
   * **Repository name:** 프로젝트 이름 (예: `my-portfolio`)을 입력합니다. (GitHub Pages URL에 영향을 미칩니다.)
   * **Description (선택 사항):** 프로젝트 설명을 입력합니다.
   * **Public/Private:** "Public"으로 설정해야 GitHub Pages에서 접근할 수 있습니다.
   * "Add a README file"은 **체크하지 않습니다.** (나중에 로컬 파일로 덮어쓸 것이기 때문에)
   * "Create repository" 버튼을 클릭합니다.

4. **로컬 프로젝트를 Git 저장소로 초기화:**
   * VS Code의 터미널 (Ctrl + `) 또는 별도의 명령 프롬프트를 열고, `my-portfolio` 프로젝트 폴더로 이동합니다.
   * 다음 Git 명령어를 순서대로 실행합니다.

   ```bash
   git init                # 현재 폴더를 Git 저장소로 초기화
   git add .               # 모든 파일 추적 (스테이징)
   git commit -m "Initial portfolio website"  # 변경사항 커밋
   git branch -M main      # 기본 브랜치 이름을 main으로 설정 (GitHub 기본값)
   git remote add origin https://github.com/YOUR_GITHUB_USERNAME/my-portfolio.git
# 원격 저장소 연결 (YOUR_GITHUB_USERNAME을 본인 GitHub 계정으로 변경)
   ```

```
git push -u origin main     # GitHub로 코드 푸시 (처음 푸시 시 사용자명/비밀번호 또는 토큰 입력)
```

* `git push` 명령 실행 시, GitHub 자격 증명을 요구할 수 있습니다. 개인 액세스 토큰(Personal Access Token, PAT)을 사용하는 것이 권장됩니다. (GitHub 설정 > Developer settings > Personal access tokens 에서 생성 가능)

5. **GitHub Pages 활성화:**
   * GitHub 저장소 페이지로 이동합니다.
   * "Settings" 탭을 클릭합니다.
   * 왼쪽 사이드바에서 "Pages"를 클릭합니다.
   * "Branch" 섹션에서 "None" 대신 "main" (또는 다른 기본 브랜치 이름)을 선택하고, 폴더는 `/(root)`로 유지합니다.
   * "Save" 버튼을 클릭합니다.
   * 몇 분 후, 페이지 상단에 웹사이트 URL이 표시됩니다. (예: `https://YOUR_GITHUB_USERNAME.github.io/my-portfolio/`)
   * 해당 URL로 접속하여 자신의 포트폴리오 웹사이트가 성공적으로 배포되었는지 확인합니다.

16시간: 검토, Q&A 및 다음 단계

이 시간은 HTML, CSS, JavaScript 전반에 걸쳐 학습한 모든 핵심 개념에 대한 포괄적인 과정 검토 및 요약을 진행합니다. 이어서 남아있는 질문에 대한 공개 Q&A 세션, 지속적인 학습을 위한 자료(예: MDN Web Docs, 온라인 커뮤니티, 고급 과정/프레임워크)를 제공하고, 더 넓은 웹 개발 로드맵에 대한 개요를 제시합니다. 기초 지식 강화, 지속적인 학습 마인드 함양, 웹 개발 생태계의 광대함을 이해하는 것이 핵심 개념입니다. 공개 토론, 자율 학습을 위한 간단한 프로젝트 아이디어 브레인스토밍, 그리고 더 복잡한 주제에 접근하는 방법에 대한 지침을 제공하는 실습을 진행합니다.

핵심 개념:

* **HTML, CSS, JavaScript 핵심 개념 요약** 및 상호 관계 재정립
* **웹 개발 과정에 대한 질의응답**
* **지속적인 학습의 중요성** 및 마인드셋
* **웹 개발 생태계 로드맵** 소개:
   * 프런트엔드 프레임워크 (**React, Vue, Angular**)
   * 백엔드 기술 (**Node.js, Python/Django, Ruby on Rails** 등)

    * 데이터베이스 (**SQL, NoSQL**)
    * 클라우드 컴퓨팅 (**AWS, Azure, GCP**)
    * 컴퓨터 과학 기초 (알고리즘, 자료구조)
    * DevOps, 테스트 등

실습/활동 예제:

1. **전체 과정 복습 및 요약 (강사 주도):**
    * **HTML**: 구조, 시맨틱 태그, 폼, 테이블
    * **CSS**: 스타일링, 박스 모델, Flexbox, 미디어 쿼리
    * **JavaScript**: 상호작용, 변수, 조건문, 반복문, 함수, 배열, DOM 조작, 이벤트
    * 각 모듈에서 배운 핵심 내용과 그것들이 어떻게 조화를 이루어 웹 페이지를 만드는지 다시 한번 강조합니다.

2. **Q&A 세션:**
    * 자유로운 질문과 답변 시간을 가집니다. 학습자들이 궁금해하는 점, 어려웠던 점 등을 해결해 줍니다.

3. **자율 학습을 위한 프로젝트 아이디어 브레인스토밍:**
    * 학습자들이 혼자서 시도해 볼 수 있는 간단한 웹 프로젝트 아이디어를 함께 생각해봅니다.
        * 예: 간단한 계산기 (JS 로직 추가), 온도 변환기, 타이머/스톱워치, 이미지 갤러리 (JS로 동적 추가), 반응형 블로그 레이아웃, 카드 게임 (JS로 카드 섞기/뒤집기), 간단한 투표 앱 등.
    * "이 프로젝트를 만들려면 HTML, CSS, JavaScript 중 어떤 부분이 필요할까요?" 와 같은 질문으로 토론을 유도합니다.

4. **다음 학습 단계 및 자료 안내:**
    * **공식 문서**: **MDN Web Docs** (가장 중요!)
    * **온라인 학습 플랫폼**: Codecademy, freeCodeCamp, Udemy, Coursera 등
    * **프런트엔드 프레임워크 소개**:
        * **React**: Facebook에서 개발, UI 컴포넌트 기반, 대규모 애플리케이션에 적합.
        * **Vue.js**: 점진적 채택 가능, 배우기 쉽고 유연함.
        * **Angular**: Google에서 개발, 풀 스택 프레임워크, 대규모 엔터프라이즈 애플리케이션에 적합.
    * **백엔드 기술 간략 소개**:
        * **Node.js** (JavaScript 기반 서버)
        * **Python** (**Django, Flask** 프레임워크)
        * **Ruby** (**Ruby on Rails** 프레임워크)

    * **버전 관리 시스템 심화**: Git 및 GitHub 심화 학습
    * **디자인 툴**: Figma, Adobe XD (간략히)
    * **알고리즘 및 자료구조**: 개발자로서 문제 해결 능력을 향상시키기 위한 중요성 강조
    * **꾸준히 코딩하고 프로젝트를 만들어 볼 것의 중요성 강조.**

5. **동기 부여 및 격려:**
    * 단기간에 모든 것을 마스터할 수는 없지만, 이 과정이 강력한 시작점임을 강조합니다.
    * 끈기와 호기심을 가지고 계속 학습하면 멋진 웹 개발자가 될 수 있음을 격려합니다.
    * 질문하는 것을 두려워하지 말고, 커뮤니티의 도움을 적극적으로 활용하라고 조언합니다.