03. 자바스크립트로 UI 업데이트하기

이 챕터에서는 **JavaScript**와 **DOM 메서드**를 사용해서 프로젝트에 `h1` 태그를 추가해보겠습니다.

우선 코드 에디터를 열고 `index.html` 파일을 새로 만듭니다. 파일 안에 다음 코드를 추가하세요:

```
<html>
  <body>
    <div></div>
  </body>
</html>
```

이제 `div`에 나중에 접근할 수 있도록 고유한 `id`를 추가합니다.

```
<html>
  <body>
    <div id="app"></div>
  </body>
</html>
```

HTML 파일 안에 JavaScript를 작성하려면 `script` 태그를 추가해야 합니다:

```
<html>
  <body>
    <div id="app"></div>
    <script type="text/javascript"></script>
  </body>
</html>
```

이제 `script` 태그 안에서 DOM 메서드인 `getElementById()`를 사용해서 `div` 요소를 선택할 수 있습니다:

```
<html>
  <body>
    <div id="app"></div>
    <script type="text/javascript">
      const app = document.getElementById('app');
    </script>
  </body>
</html>
```

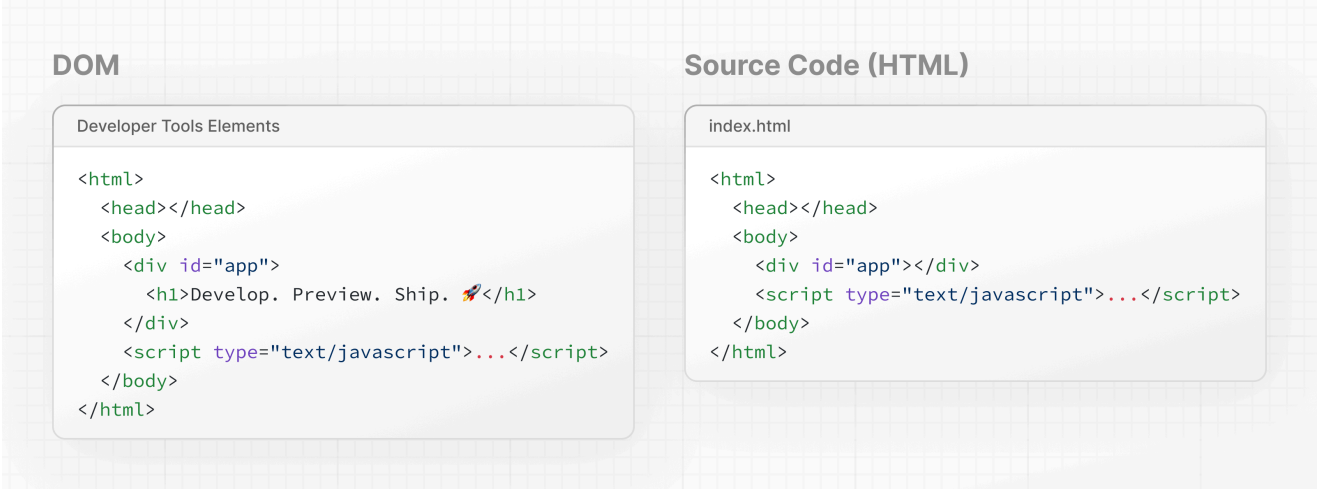
계속해서 DOM 메서드를 사용해 새로운 `<h1>` 요소를 생성합니다:

```html
<html>
  <body>
    <div id="app"></div>
    <script type="text/javascript">
      // 'app' id를 가진 div 요소 선택
      const app = document.getElementById('app');

      // 새 H1 요소 생성
      const header = document.createElement('h1');

      // H1 요소에 넣을 텍스트 노드 생성
      const text = 'Develop. Preview. Ship.';
      const headerContent = document.createTextNode(text);

      // 텍스트를 H1 요소에 추가
      header.appendChild(headerContent);

      // H1 요소를 div 안에 추가
      app.appendChild(header);
    </script>
  </body>
</html>
```

모든 작업이 제대로 되었는지 확인하려면 브라우저에서 HTML 파일을 열어보세요. 'Develop. Preview. Ship.' 이라는 글자가 있는 `h1` 태그가 보일 것입니다.

HTML vs. DOM

브라우저의 개발자 도구에서 DOM 요소를 살펴보면, `<h1>` 요소가 DOM 안에 포함되어 있는 것을 확인할 수 있습니다.



> (이미지 링크: 위)

이는 HTML이 **초기 페이지 내용**을 나타내고, DOM은 **자바스크립트로 변경된 현재 페이지 상태**를 나타내기 때문입니다.

순수 JavaScript로 DOM을 업데이트하는 것은 강력하지만 코드가 길어집니다. 방금 `h1` 요소 하나를 추가하는 데 이렇게 많은 코드를 작성했습니다:

```
<script type="text/javascript">
  const app = document.getElementById('app');
  const header = document.createElement('h1');
  const text = 'Develop. Preview. Ship.';
  const headerContent = document.createTextNode(text);
  header.appendChild(headerContent);
  app.appendChild(header);
</script>
```

애플리케이션이나 팀 규모가 커지면 이런 방식은 점점 비효율적이 됩니다.

개발자는 "무엇을 보여줄지"가 아니라 "어떻게" 보여줄지를 일일이 명령해야 합니다. 그런데 만약 "무엇을 보여주고 싶은지"만 선언하고, 나머지는 컴퓨터가 알아서 처리해주면 좋지 않을까요?

명령형 프로그래밍 vs 선언형 프로그래밍

위 코드는 **명령형(Imperative)** 프로그래밍의 좋은 예입니다. 즉, UI를 어떻게 업데이트할지를 한 단계씩 지시합니다.

하지만 UI를 구축할 때는 **선언형(Declarative)** 접근 방식을 선호하는 경우가 많습니다. 선언형 방식에서는 DOM 메서드를 일일이 다루는 대신 "어떤 것을 보여주고 싶은지"만 기술합니다. 예를 들어 `h1` 태그에 텍스트를 넣고 싶다고 선언하는 것이죠.

비유를 들자면:

- 명령형: 셰프에게 피자 만드는 모든 과정을 하나하나 지시한다
- 선언형: "피자 하나 주세요"라고 주문만 한다

React: 선언형 UI 라이브러리

React는 이런 **선언형** 프로그래밍 방식을 지원하는 인기 라이브러리입니다.

개발자는 "UI에 무엇을 표시하고 싶은지"만 React에게 알려주면, React가 내부적으로 DOM 업데이트 방법을 결정해줍니다.

다음 섹션에서는 React를 어떻게 시작할 수 있는지 배워봅니다.