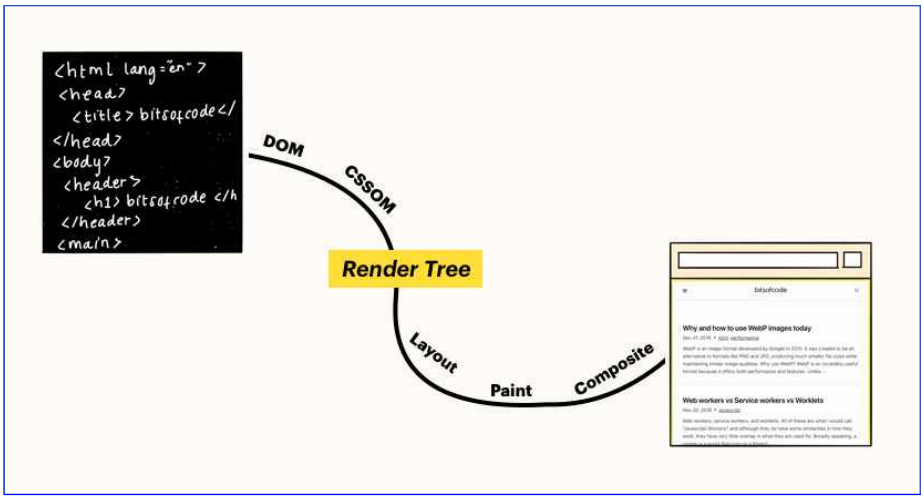
DOM(Document Object Model)

☑ DOM은 정확히 무엇일까?

DOM(Document Object Model)은 Web Page에 대한 Interface 이다. 기본적으로 여러 프로그램들이 Page의 콘텐츠 및 구조, 그리고 style을 읽고 조작할 수 있도록 API를 제공한다. 먼저 DOM을 이해하기 전에 Web Page가 어떻게 빌드 되는지 살펴본다.

☑ Web Page는 어떻게 만들어질까?

Web Browser가 원본 HTML document를 읽어들인 후, style을 입히고 대화형 Page로 만들어 view port에 표시하기까지의 과정을 "Critical Rendering Path"라고 한다. Critical Rendering Path 과정은 여러 단계로 나누어져 있지만, 이 단계들을 대략 두 단계로 나눌 수 있다. 첫 번째 단계에서 Browser는 읽어들인 document를 파싱하여 최종적으로 어떤 내용을 Page에 rendering 할지 결정한다. 그리고 두 번째 단계에서 Browser는 해당 rendering을 수행한다.



첫 번째 과정을 거치면 "render tree"가 생성된다.
render tree는 Web Page에 표시될 HTML element들과 이와 관련된 style element들로 구성된다. Browser는 render tree를 생성하기 위해 다음과 같이 두 모델이 필요하다.

☞ DOM(Document Object Model) - HTML element들의 구조화된 표현
☞ CSSOM(Cascading Style Sheets Object Model) - element들과 연관된 style 정보의 구조화된 표현

☑ DOM은 어떻게 생성될까 (그리고 어떻게 보여질까)?

DOM은 source HTML document의 object-based 표현 방식이다. 둘은 서로 비슷하지만, DOM이 갖고 있는 근본적인 차이는 단순 텍스트로 구성된 HTML document의 내용과 구조가 객체 모델로 변환되어 다양한 프로그램에서 사용될 수 있다는 점이다.
DOM의 개체 구조는 "node tree"로 표현된다. 하나의 parent stem(부모 줄기)가 여러 개의 child branches(자식 나뭇가지)를 갖고 있고, 또 각각의 나뭇가지는 leave(잎)들을 가질 수 있는 나무와 같은 구조로 이루어져 있기 때문이다. 이 경우, parent "stem"은 root <html> element이고, child "branches"는 중첩된 element 이며, "leaves"은 element 내의 컨텐츠이다.

아래 HTML 예시에서,

```
<!doctype html>
<html lang="en">
 <head>
   <title>My first web page</title>
  </head>
 <body>
    <h1>Hello, world!</h1>
    <p>How are you?</p>
  </body>
</html>
```

이 document는 아래와 같은 node tree로 표현된다.

☑ DOM이 아닌 것

위 예제 혹은 DevTools에서 DOM은 마치 HTML document와 1 대 1 매핑이 되는 것처럼 보인다. 그러나 둘 간에는 몇 가지 차이점이 있다.

DOM이 무엇인지 완전히 이해하기 위해, 어떤 것이 DOM이 아닌지 살펴본다.

1. DOM은 HTML이 아니다.

DOM은 HTML document로부터 생성되지만 항상 동일하지 않다. DOM이 원본 HTML 소스와 다를 수 있는 두 가지 케이스가 있다.

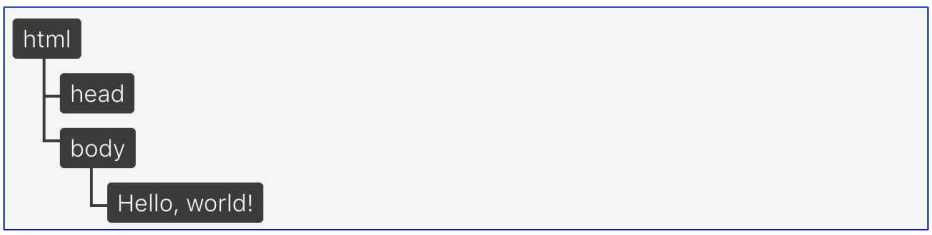
☞ 작성된 HTML document가 유효하지 않을 때

DOM은 유효한 HTML document의 인터페이스 이다. DOM을 생성하는 동안, Browser는 유효하지 않은 HTML 코드를 올바르게 교정한다.

아래 예시를 살펴보면,

```
<!doctype html>
<html>
Hello, world!
</html>
```

document에 유효한 HTML 규칙의 필수 사항인 <head> 와 <body> element가 빠져있다. 그렇지만 생성된 DOM 트리에는 올바르게 교정되어 나타난다.



☞ JavaScript에 의해 DOM이 수정될 때

DOM은 HTML document의 내용을 볼 수 있는 인터페이스 역할을 하는 동시에 동적 자원이 되어 수정될 수 있다. 예를 들어, JavaScript를 사용해 DOM에 새로운 node를 추가할 수 있다.

```
var newParagraph = document.createElement("p");
var paragraphContent = document.createTextNode("I'm new!");
newParagraph.appendChild(paragraphContent);
document.body.appendChild(newParagraph);
```

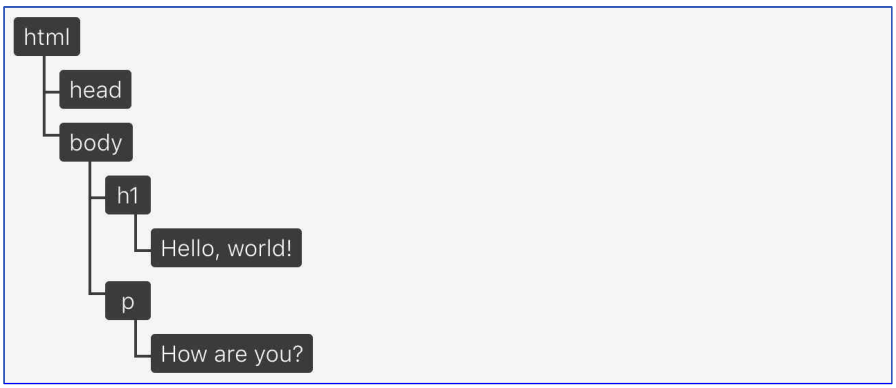
이 코드는 DOM을 업데이트한다. 하지만 HTML document의 내용을 변경하진 않는다.

2. DOM은 Browser에서 보이는 것이 아니다.

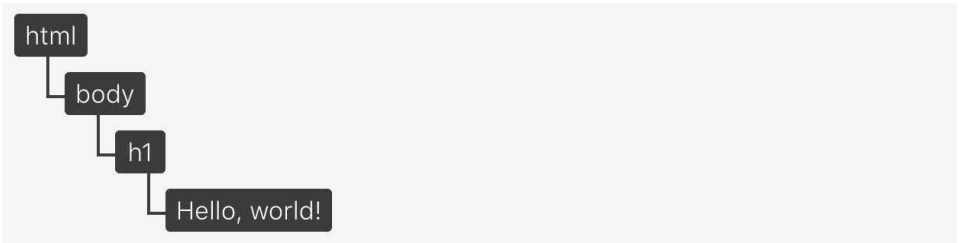
Browser view port에 보이는 것은 render tree로 DOM과 CSSOM의 조합이다. render tree는 오직 스크린에 그려지는 것으로 구성되어 있어 DOM과 다르다.
달리 말하면, rendering 되는 element 만이 관련 있기 때문에 시각적으로 보이지 않는 element는 제외e 된다. 예를 들어, display: none style 속성을 가지고 있는 element 이다.

```
<!doctype html>
<html lang="en">
  <head></head>
  <body>
    <h1>Hello, world!</h1>
    <p style="display: none;">How are you?</p>
  </body>
</html>
```

DOM은 <p> element를 포함시킨다.



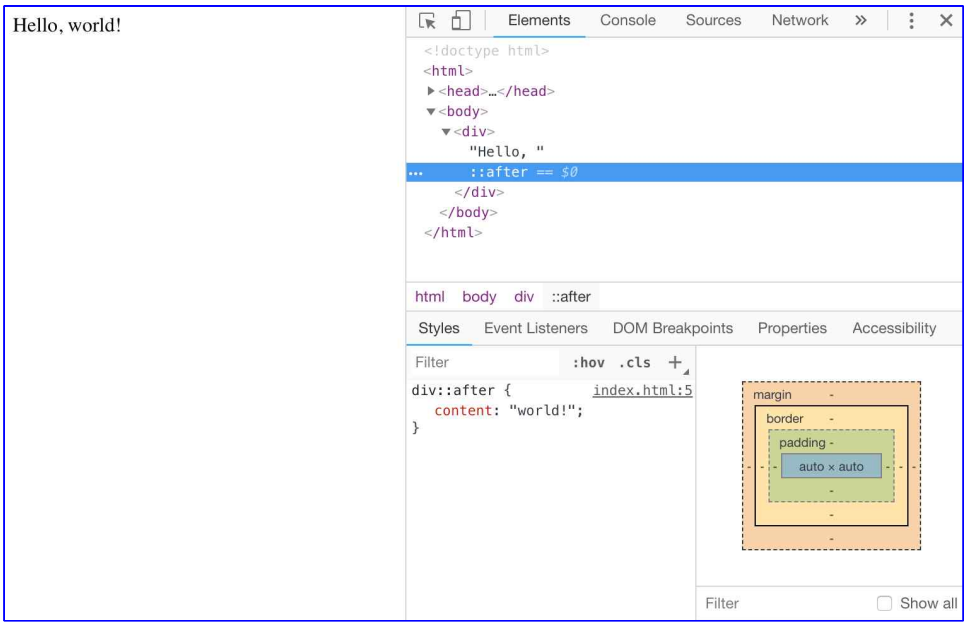
그러나 렌더 트리에 해당하는 view port에 표시되는 내용은 <p> element를 포함하지 않는다.



3. DOM은 개발도구에서 보이는 것이 아니다.

개발도구의 element 검사기는 DOM과 가장 가까운 근사치를 제공한다. 그러나 개발도구의 element 검사기는 DOM에 없는 추가적인 정보를 포함한다. 가장 좋은 예는 CSS의 가상 element 이다.
::before 과 ::after 선택자를 사용하여 생성된 가상 element는 CSSOM과 render tree의 일부를 구성한

다.
그러나 기술적으로 DOM의 일부는 아니다. DOM은 오직 원본 HTML document로부터 빌드되고, element에 적용되는 style을 포함하지 않기 때문이다.
가상 element가 DOM의 일부가 아님에도 불구하고, element 검사기에서는 아래와 같이 확인된다.



이러한 이유로 가상 element는 DOM의 일부가 아니기 때문에 JavaScript에 의해 수정될 수 없다.

정리하면 DOM은 HTML document에 대한 인터페이스 이다. 첫째로 view port에 무엇을 rendering 할지 결정하기 위해 사용되며, 둘째로는 Page의 콘텐츠 및 구조, 그리고 style이 JavaScript 프로그램에 의해 수정되기 위해 사용된다.

DOM은 원본 HTML document 형태와 비슷하지만 몇 가지 차이점이 있다.

- ☞ 항상 유효한 HTML 형식이다.
- ☞ JavaScript에 수정될 수 있는 동적 모델이어야 한다.
- ☞ 가상 element를 포함하지 않는다. (Ex. ::after)
- ☞ 보이지 않는 element를 포함한다. (Ex. display: none)