

Session 11

JavaScript DOM

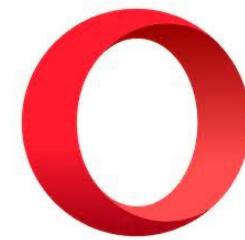
NEXT 9기 운영진 강단비

JS 들어가며

어떻게 확장자가 다른 파일들이 서로를 이해할 수 있을까?

```
1  <!DOCTYPE html>
2  <html lang="en">
3  |  <head>
4  |  |  <meta charset="UTF-8" />
5  |  |  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6  |  |  <title>Document</title>
7  |  |  <link rel="stylesheet" sype="text/css" href="style.css" />
8  |  </head>
9  |  <body>
10 |  |  <script type="text/javascript" src="script.js"></script>
11 |  </body>
12 </html>
```

Web Browser



DOM은 이러한 웹 브라우저에 내장되어 있는 API

Document Object Model

Document(=html)을 Object로 Modeling 했다 (?)

= Html을 JS가 이해할 수 있는 모양으로 만들었다

브라우저가 만든 DOM은 node, property, method로 구성

JS

DOM

DOM이란?

NEXT[9].강단비

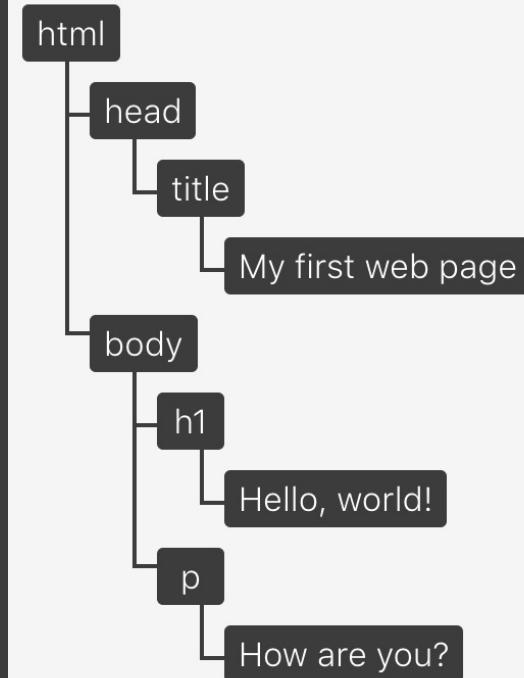
CSSOM 역시 존재합니다

(더보기)

https://developer.mozilla.org/ko/docs/Web/API/CSS_Object_Model

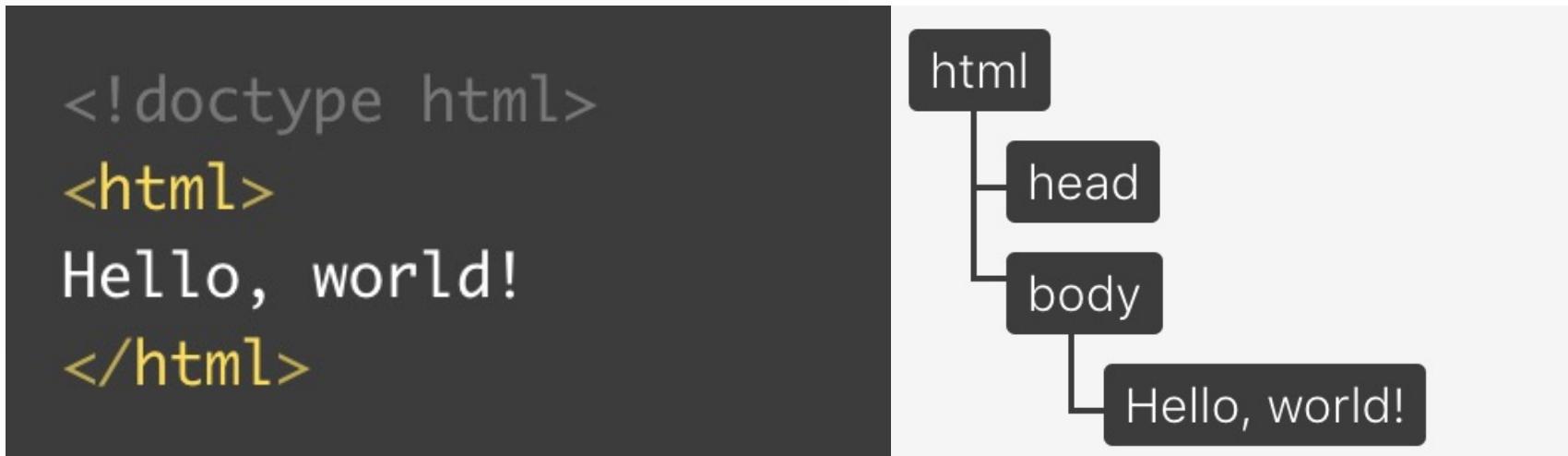
JS DOM이란?

```
<!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>
```



1. DOM != HTML

작성된 HTML 문서가 유효하지 않을 때



DOM은 유효한 HTML 문서의 인터페이스임
DOM을 생성하는 동안 브라우저는 유효하지 않은 HTML 코드를 유효하게 교정함.

JS DOM이란?

1. DOM != HTML

자바스크립트에 의해 DOM이 수정될 때

```
let newDiv = document.createElement("div");
let divContent = document.createTextNode("Hello, I'm new div");

newDiv.appendChild(divContent);
document.body.appendChild(newDiv);
```

이 코드는 DOM을 업데이트하긴 하지만,
HTML 문서의 내용을 변경하지는 않음.

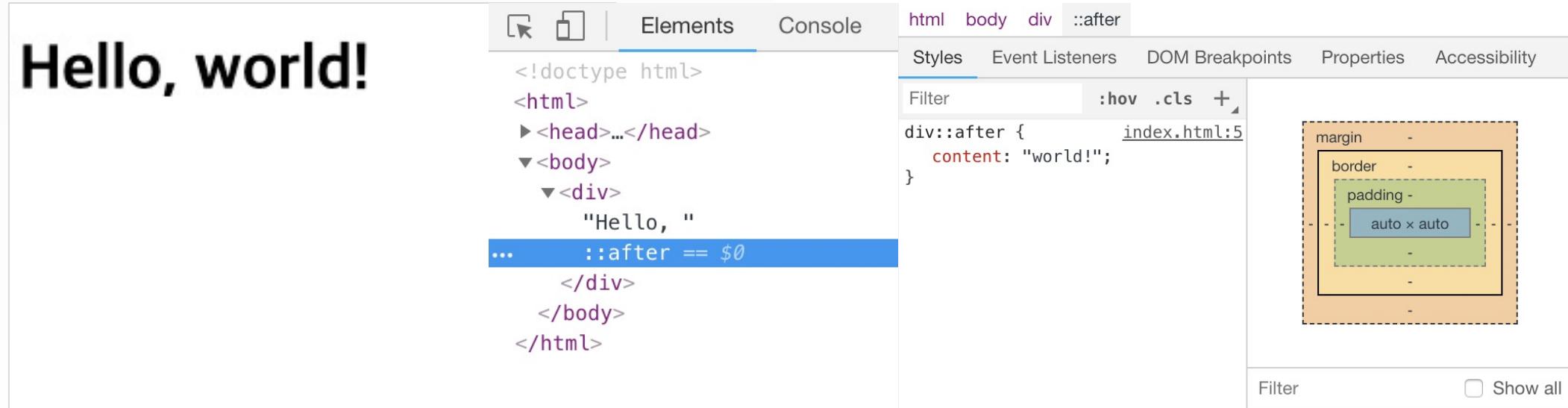
2. DOM != 브라우저 뷰 포트

Hello, world!

```
<!DOCTYPE html>
<html lang="en">
  <head> </head>
  <body>
    <h1>Hello, world!</h1>
    <p style="display: none">피곤해~</p>
  </body>
</html>
```

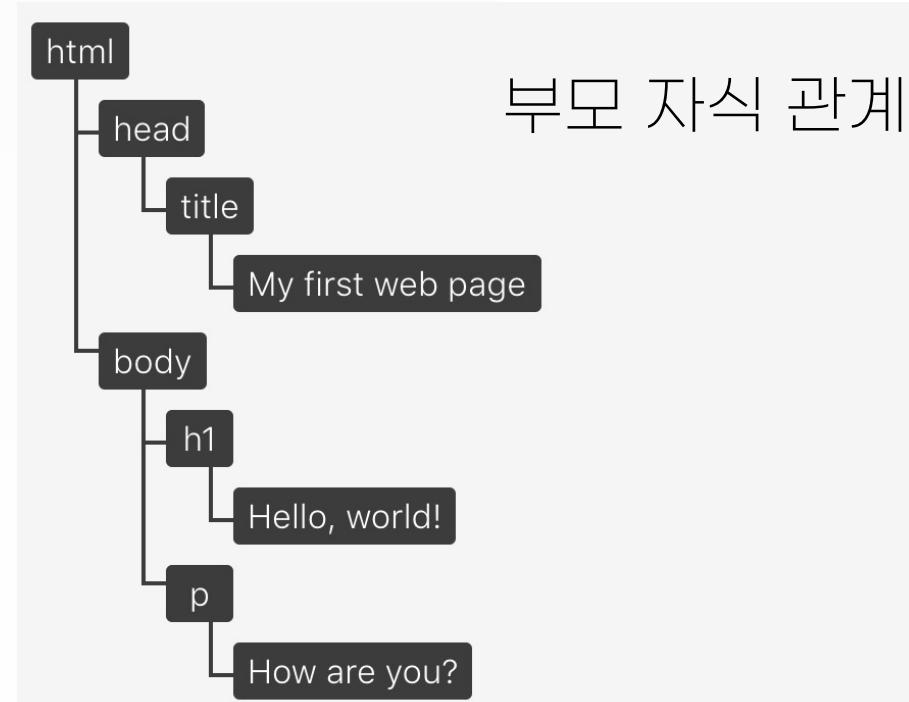
브라우저 뷰 포트에 보이는 것은 **렌더 트리**(DOM과 CSSOM의 조합)
렌더링되는 요소만이 관련 있기 때문에 시각적으로 보이지 않는 요소는 제외됨

3. DOM != 개발자도구의 elements



개발자도구의 elements는 DOM과 가장 가까운 모습.
하지만 이는 DOM에는 없는 CSS의 가상 요소인 추가적인 정보까지 포함.

노드는 계층적 구조!



DOM 요소에 접근하려면!

| DOM 조작 실습 – DOM 고르기

```
<body>
  <article>
    <h1 id="title" class="title">자스 어려워요.</h1>
  </article>
  <script type="text/javascript">
    const title1 = document.getElementById('title');

    const title2 = document.querySelector('#title');
    const title3 = document.querySelector('.title');

    console.log(title1 === title2); //true;
    console.log(title1, title2, title3);
  </script>
</body>
```

document.getElementById

- DOM의 Elements를 가지고 id로 가지고 오겠다

document.querySelector (첫번째 원소)

- Css 선택자를 통해 DOM의 Element를 가지고 오겠다.

document.querySelectorAll (배열)

- Css 선택자를 통해 DOM의 Element"s"를 가지고 오겠다.

getElementsByClassName, getElementsByTagName등 다양하지만 다 몰라도된다.

JS index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
    <link rel="stylesheet" type="text/css" href="style.css" />
  </head>
  <body>
    <div class="parent">
      <p id="son1" class="child">아들1</p>
      <p id="son2" class="child">아들2</p>
    </div>
    <script type="text/javascript" src="script.js"></script>
  </body>
</html>
```

```
.red {
  color: red;
```

JS querySelector

```
const parent = document.querySelector(".parent");
const son1 = document.querySelector("#son1");
const son2 = parent.querySelector("#son2");
```

JS console에 찍어보자

해당 요소에 대한 정보를 얻고 싶을 때!

console.log

요소를 HTML과 같은
트리 구조로 출력

console.dir

요소를 JSON과 같은
트리 구조로 출력

그 외

```
> console.info('정보')  
정보
```

```
> console.warn('경고')  
⚠▶ 경고
```

```
> console.error('에러')  
✖▶ 에러
```

JS classList

```
const parent = document.querySelector(".parent");
const son1 = document.querySelector("#son1");
const son2 = parent.querySelector("#son2");

son1.classList.add("red");
// son1.classList.remove("red");
// son1.classList.toggle("red");
```

나머지 자식도 바꿔봅시다!

JS Activity

```
const parent = document.querySelector(".parent");
const son1 = document.querySelector("#son1");
const son2 = parent.querySelector("#son2");

son1.classList.add("red");
son2.classList.add("red");
```

하지만 자식이 겁나 많다면?

JS Activity

```
const parent = document.querySelector(".parent");
const son1 = document.querySelector("#son1");
const son2 = parent.querySelector("#son2");
const children = parent.querySelectorAll(".child");

children.classList.add("red");
```

왜 안될까요?

console에 찍어봅시다

나머지 자식들도 바꿔주세요!

JS

textContent / innerText / innerHTML

```
<p id="son1" class="child">저는<b style="display: none">자식입니다.</b></p>
```

textContent

가급적 `textContent`를 사용하는 것이 좋습니다. 성능과 보안에 강점이 있고, 결과적으로 해당 노드의 raw text를 얻게 됨으로써 이후 의도한 대로 가공할 수 있기 때문입니다.

console에 찍어봅시다

innerText

특정 노드에 렌더링 된(화면에 보이는 그대로의) 텍스트를 읽어올 때 유용합니다. 하지만 내부에 특별히 스타일 적용이 없는 문자열을 할당할 때는 성능상 적합하지 않습니다. 단, IE(IE8 이하) 환경을 중점으로 고려한 프로젝트라면 `textContent` 대신 사용하도록 합니다. 😊

innerHTML

HTML Parsing이 필요한 문자열에만 사용하도록 합니다. 그게 아니라면, 성능상 좋지 않고 XSS 공격에도 취약하므로 `innerHTML`은 사용하지 않는 것이 좋습니다.

JS **textContent / innerText / innerHTML**

내용을 바꿔봅시다

```
son1.textContent = "저는 아들1 입니다.";  
son2.textContent = "저는 아들2 입니다."  
  
son1.innerHTML = "<b>저는</b> 아들입니다.";
```

createElement() / appendChild()

“ 예나, 선정이 딸이에요 ” 를 구현해보자.



JS

createElement() / appendChild()

```
const seonjeong = document.createElement("div")
const yena = document.createTextNode("옹애")

seonjeong.appendChild(yena)
document.body.appendChild(seonjeong)
```

JS setAttribute

```
seonjeong.setAttribute("id", "seonjeong");
seonjeong.setAttribute("class", "parent");
yena.setAttribute("id", "yena");
yena.setAttribute("class", "child");
```

JS Activity 2

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <a href="https://www.naver.com">네이버로 가기</a>
    

    <script>
      document.querySelector('a').href = 'https://www.daum.net';
      document.querySelector('img').src = 'https://picsum.photos/200/200';
    </script>
  </body>
</html>
```

- attribute.html 을 따로 만들고,
- 아래와 같이 네이버 링크와 이미지 삽입.
- 스크립트 태그 내에서 링크를 구글로 변경
- 사진은 다른 사진으로 변경

addEventListener

<https://developer.mozilla.org/ko/docs/Web/Events>

사용자의 액션에 반응하는 화면을 만들어봅시다

- 새로운 Html 파일과 JS파일을 만들어주세요.

JS addEventListener

```
const title = document.querySelector(".title");

title.addEventListener("mouseover", function (event) {
  console.log(event);
});

title.addEventListener("mouseout", function (event) {
  console.log(event);
});
```

JS Event.target

```
const title = document.querySelector(".title");
const GOLD = "#f1c40f";
const GRAY = "#34495e";

const onMouseOver = (event) => (event.target.style.color = GOLD);
title.addEventListener("mouseover", onMouseOver);

const onMouseOut = (event) => (event.target.style.color = GRAY);
title.addEventListener("mouseout", onMouseOut);
```

이번엔 클릭을 잡아봅시다

```
<h1 class="title">자바스크립트는 근본이 없군요!</h1>
<button>그렇다면,</button>
```

JS click

```
const title = document.querySelector(".title");
const button = document.querySelector("button");

const changedText = "한주님과 함께 TypeScript를 배워봅시다.';

const onClick = (event) => (title.textContent = changedText);
button.addEventListener("click", onClick);
```

JS Activity 3

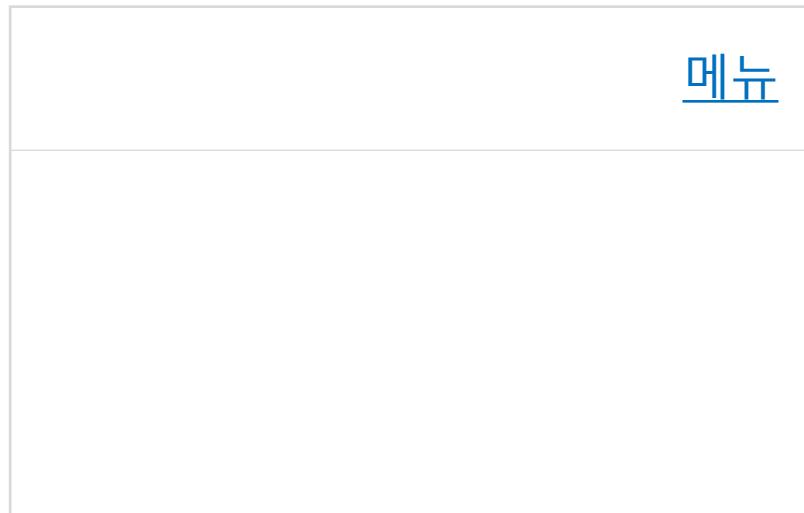
```
<h1>버튼을 바꿔보자</h1>
<button>눌러주세요</button>
<button>눌러주세요</button>
<button>눌러주세요</button>
<button>눌러주세요</button>
```

- 이와 같은 html 만들어주세요
- 버튼을 눌렀을 경우 해당 버튼의 텍스트가
‘짜잔’으로 바뀌게 하세요.

JS Assignment

Issue 1 블로그의 새 글 작성/수정 페이지에서 글자 수 표시

Issue 2 블로그 메뉴를 아래와 같이 토글 가능하도록 수정



메뉴



접기

홈으로

로그인
(그외...)