1.웹페이지 제작 실습(개인 > 팀)
2.자바스크립트 > 타입스크립트
3.DOM 요소 자세히

실습을 하기 위해서는 DOM 요소에 대해 자세히 알아야 함

DOM + 간단한 웹페이지 제작 ➡ 리액트

DOM, EVENT

1. 노드 접근하기

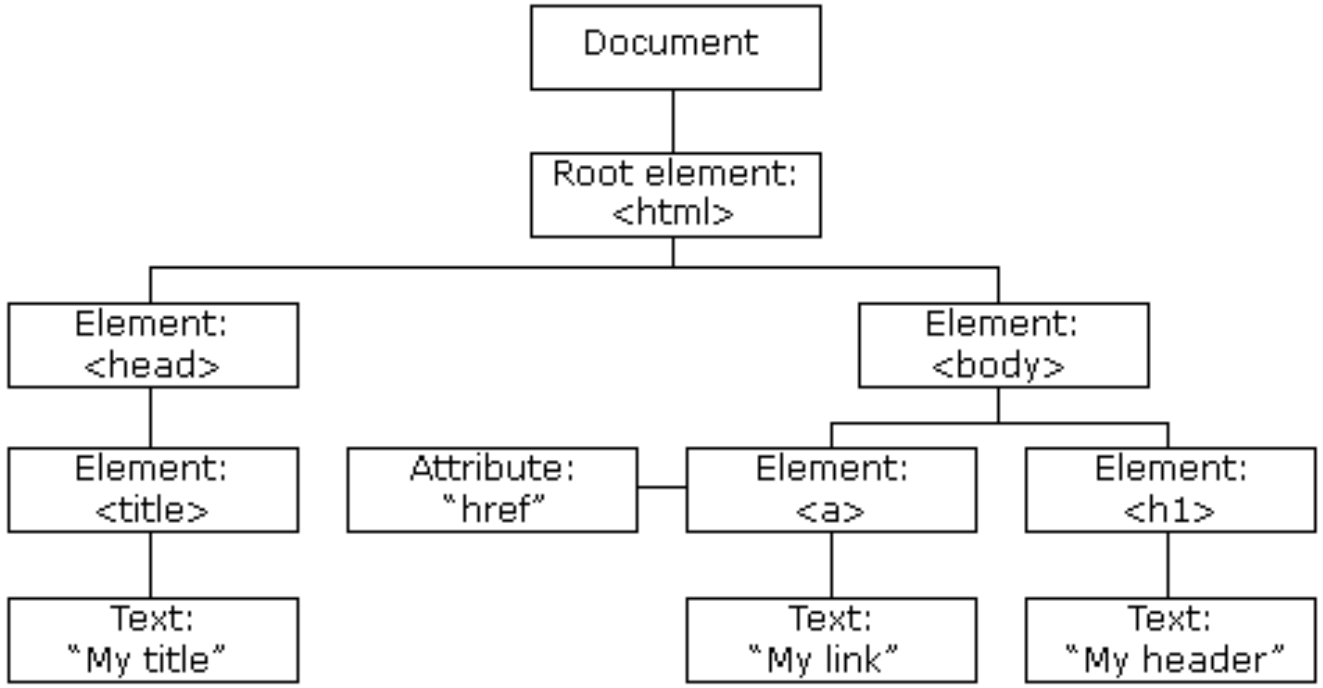
Node

- HTML DOM은 노드(Node)라고 불리는 계층적 단위에 정보를 저장하고 있음
- HTML DOM은 이러한 노드들을 정의하고 그들 사이의 관계를 설명해주는 역할을 함
- 종류: 문서 노드, 요소 노드, 속성 노드, 텍스트 노드, 주석 노드

노드	설명
문서 노드	HTML 문서 전체를 나타내는 노드
요소 노드	모든 HTML 요소는 요소 노드이며, 속성 노드를 가질 수 있는 유일한 노드
속성 노드	모든 HTML 요소의 속성은 속성 노드이며, 요소 노드에 관한 정보를 가지고 있음
텍스트 노드	HTML 문서의 모든 텍스트는 텍스트 노드임
주석 노드	HTML 문서의 모든 주석은 주석 노드임

DOM

- Document Object Model
- Html 문서의 각 요소들을 트리(tree) 형식으로 표현



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>DOM</title>
</head>
<body>
    <h1>DOM</h1>
    <h2>EVENT</h2>
    <h3>1</h3>
    <p id="first" class="pTag">Lorem ipsum dolor sit amet.</p>
    <a href="https://www.naver.com/" class="link">naver</a>
    <h3>2</h3>
    <p class="pTag">Lorem ipsum dolor sit amet consectetur, adipisicing elit.</p>
    <a href="https://www.naver.com/" class="link">naver</a>
    <h3>3</h3>
    <p class="pTag">Lorem, ipsum dolor sit amet consectetur adipisicing.</p>
    <a href="https://www.naver.com/" class="link">naver</a>
</html>
```

DOM

- 모든 Html 태그들은 객체
- 객체는 자바스크립트로 제어하고 접근 가능

DOM

EVENT

1

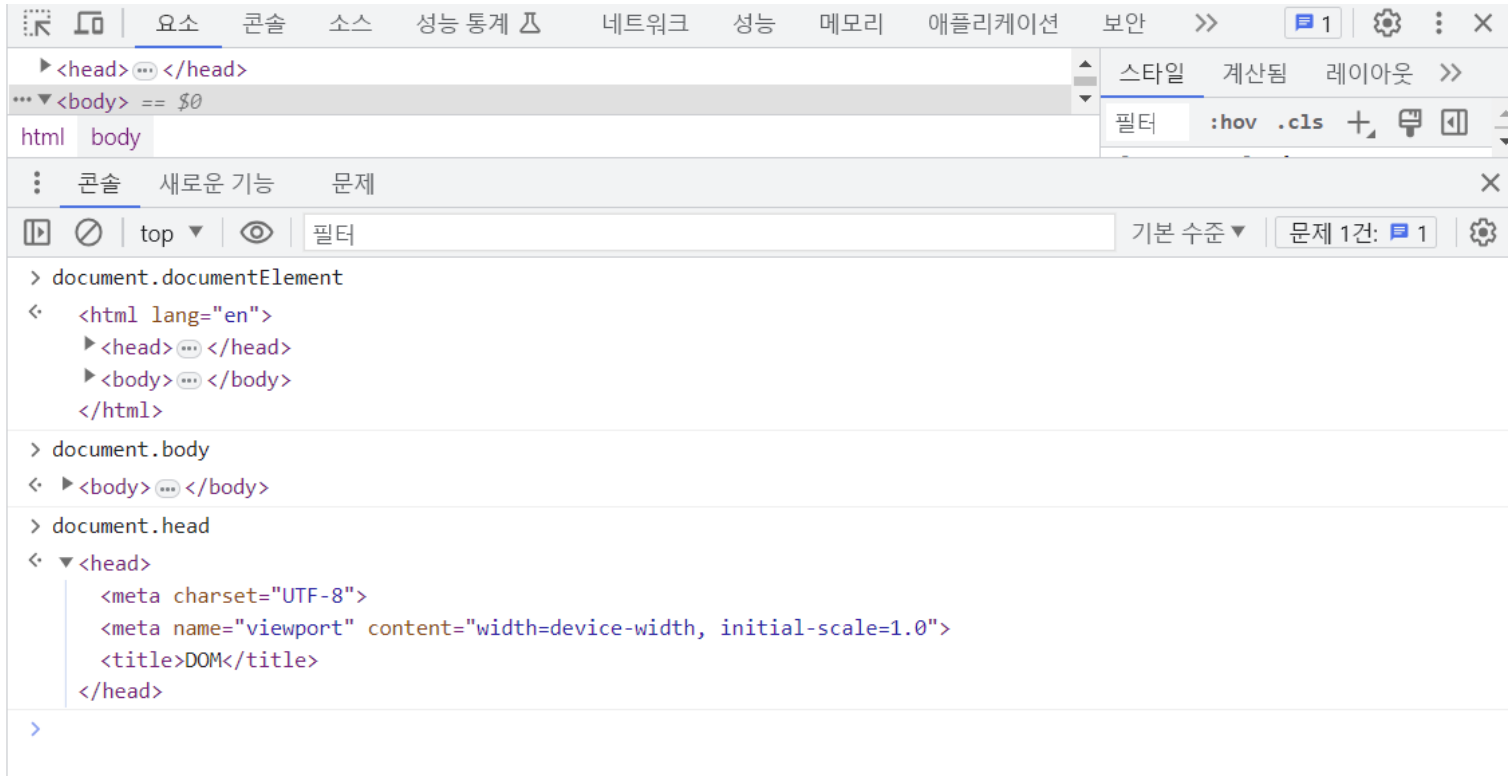
Lorem ipsum dolor sit amet consectetur adipisicing elit. Asperiores quae explicabo molestias. Omnis officia, minus similique voluptate quas ex incidunt.

2

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Facere in necessitatibus qui, quod ad labore!

3

Lorem ipsum dolor sit amet consectetur adipisicing elit. Praesentium voluptates, tempora suscipit eos quaerat nam eveniet nobis! Atque laudantium cupiditate eos odit qui, at ex!



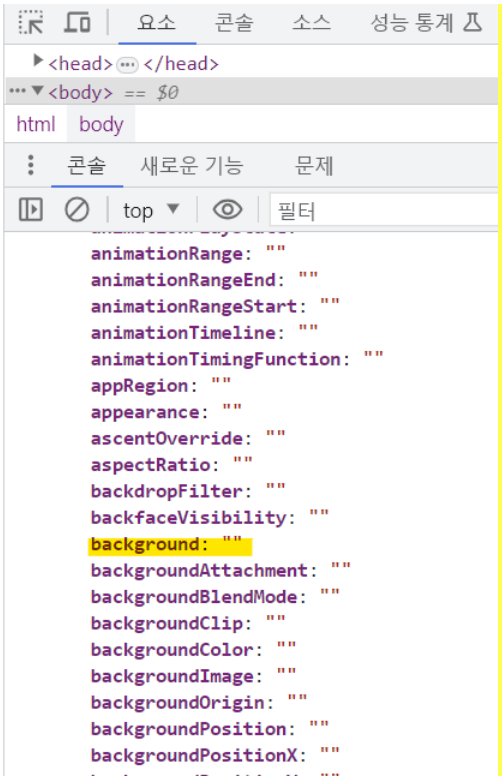
DOM

- 모든 Html 태그들은 객체
- 객체는 자바스크립트로 제어하고 접근 가능

nsectetur adipisicing elit. Asperiores quae explicabo similique voluptate quas ex incidunt.

nsectetur, adipisicing elit. Facere in necessitatibus

nsectetur adipisicing elit. Praesentium voluptates, nam eveniet nobis! Atque laudantium cupiditate eos



DOM

EVENT

1

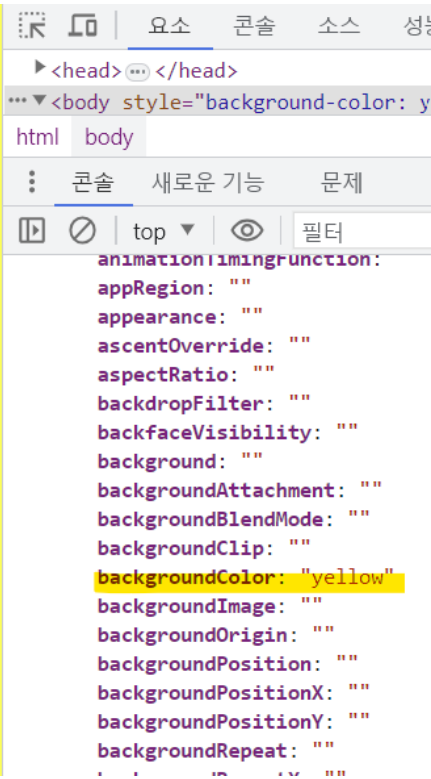
Lorem ipsum dolor sit amet consectetur adipisicing elit. Asperiores quae explicabo molestias. Omnis officia, minus similique voluptate quas ex incidunt.

2

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Facere in necessitatibus qui, quod ad labore!

3

Lorem ipsum dolor sit amet consectetur adipisicing elit. Praesentium voluptates, tempora suscipit eos quaerat nam eveniet nobis! Atque laudantium cupiditate eos odit qui, at ex!

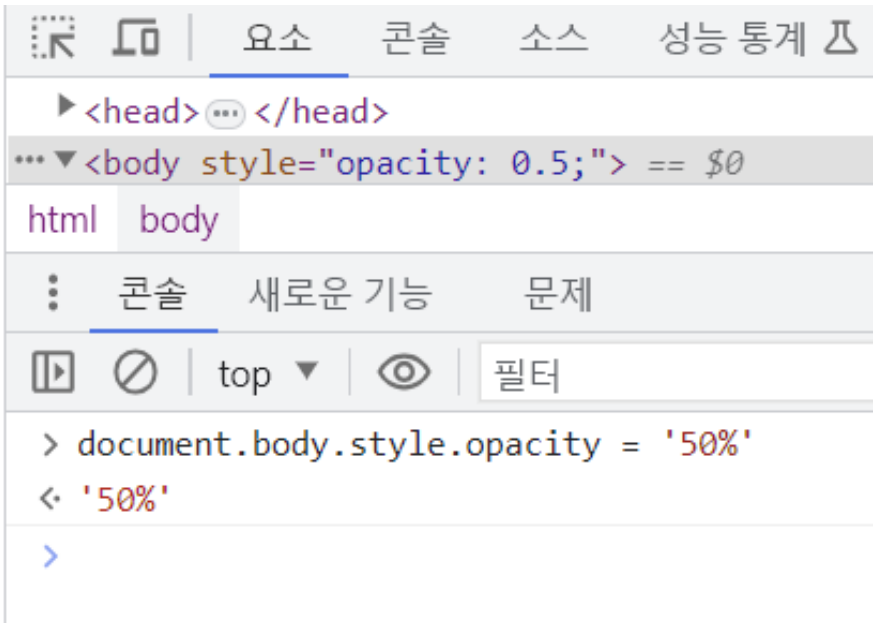


DOM

EVENT

1

Lorem ipsum dolor sit amet consectetur adipisicing elit. Asperiores quae explicabo molestias. Omnis officia, minus similique voluptate quas ex incidunt.



DOM

document.querySelector()

- 제공한 선택자/선택자 뭉치와 일치하는 문서 내 첫 번째 element 반환
- 일치하는 요소 없으면 null 반환

document.getElementById()

- 주어진 문자열과 일치하는 id 속성 가진 요소를 찾고 이를 나타내는 element 객체 반환
- id는 문서 내에서 유일해야 하기 때문에 특정 요소를 빠르게 찾을 때 유용

```
> document.querySelector('#first')
<-   <p id="first" class="pTag">Lorem ipsum dolor sit amet.</p>
> document.getElementById('first')
<-   <p id="first" class="pTag">Lorem ipsum dolor sit amet.</p>
```

DOM

document.querySelectorAll()

- 지정된 셀렉터(selector) 그룹에 일치나는 다큐먼트의 element 리스트를 나타내는 **정적 NodeList를 반환**

```
> document.querySelectorAll('.pTag')
<· ▶ NodeList(3) [p#first.pTag, p.pTag, p.pTag]
```

DOM

document.getElementsByClassName()

- 주어진 클래스(class)를 가진 모든 자식 element의 **실시간 HTML Collection을 반환**

```
> document.getElementsByClassName('pTag')
◂ ▶ HTMLCollection(3) [p#first.pTag, p.pTag, p.pTag, first: p#first.pTag]
> document.getElementsByClassName('link')
◂ ▶ HTMLCollection(3) [a.link, a.link, a.link]
```

DOM

document.getElementsByTagName()

- element의 HTML Collection과 주어진 태그명을 반환
- 루트 노드를 포함해 전체 다큐먼트를 대상으로 검색
- 자동으로 업데이트

```
> document.getElementsByTagName('p')
< ▶ HTMLCollection(3) [p#first.pTag, p.pTag, p.pTag, first: p#first.pTag]
```

DOM

```
> const pList1 = document.querySelectorAll('p');
  const pList2 = document.getElementsByTagName('p');

  pList1;
  ▶ NodeList(3) [p#first.pTag, p.pTag, p]
> pList2;
  ▶ HTMLCollection(3) [p#first.pTag, p.pTag, p, first: p#first.pTag]
```

```
▼ <body>
    <h1>DOM</h1>
    <h2>EVENT</h2>
    <h3>1</h3> <p>ppp</p>
    <p id="first" class="pTag">Lorem
    <a href="https://www.naver.com/"
```

```
> pList1;
  ▶ NodeList(3) [p#first.pTag, p.pTag, p]
> pList2;
  ▶ HTMLCollection(4) [p, p#first.pTag, p.pTag, p, first: p#first.pTag]
```

NodeList, HTML Collection

- 배열처럼 보이지만 배열❌, 유사 배열
- 이터러블
- Symbol.iterator를 프로퍼티 키로 사용한 메서드를 직접 구현하거나 프로토타입 체인을 통해 상속받은 객체
- for문 사용

```
> const pList = document.getElementsByTagName('p')
< undefined
> pList
< ▶ HTMLCollection(3) [p#first.pTag, p.pTag, p.pTag, first: p#first.pTag]
> for(p of pList){
      p.style.color = 'green'
  }
< 'green'
```

NodeList, HTML Collection

```
> document.querySelectorAll('.pTag')
◂ ▶ NodeList(3) [p#first.pTag, p.pTag, p.pTag]
> document.querySelectorAll('.link')
◂ ▶ NodeList(3) [a.link, a.link, a.link]
> const pList2 = document.querySelectorAll('p:nth-of-type(n)')
◂ undefined
> for(p of pList2){
      p.style.backgroundColor = '#000';
      p.style.color = '#fff';
  }
◂ '#fff'
```

Lorem ipsum dolor sit amet.

naver

2

Lorem ipsum dolor sit amet consectetur, adipisicing elit.

naver

3

Lorem, ipsum dolor sit amet consectetur adipisicing.

```
<p id="first" class="pTag" style="background-color: rgb(0, 0, 0); color: rgb(255,
255, 255);">Lorem ipsum dolor sit amet.</p>
<a href="https://www.naver.com/" class="link">naver</a>
<h3>2</h3>
<p class="pTag" style="background-color: rgb(0, 0, 0); color: rgb(255, 255, 255);
">Lorem ipsum dolor sit amet consectetur, adipisicing elit.</p>
<a href="https://www.naver.com/" class="link">naver</a>
<h3>3</h3>
<p class="pTag" style="background-color: rgb(0, 0, 0); color: rgb(255, 255, 255);
">Lorem, ipsum dolor sit amet consectetur adipisicing.</p>
<a href="https://www.naver.com/" class="link">naver</a>
```

html body h3

콘솔 새로운 기능 문제

top 필터

2. 부모, 자식, 형제 노드

```
<body>
    <h1>DOM</h1>
    <h2>EVENT</h2>
    <h3>1</h3>
    <p id="first" class="pTag">Lorem ipsum dolor sit amet.</p>
    <a href="https://www.naver.com/" class="link">naver</a>
    <h3>2</h3>
    <p class="pTag">Lorem ipsum dolor sit amet consectetur, adipisicing elit.</p>
    <a href="https://www.naver.com/" class="link">naver</a>
    <h3>3</h3>
    <p class="pTag">Lorem, ipsum dolor sit amet consectetur adipisicing.</p>
    <a href="https://www.naver.com/" class="link">naver</a>
    <ul id="color">
        <li id="red">red</li>
        <!-- 노드 연습 중 -->
        <li id="blue">blue</li>
        <li id="green">green</li>
    </ul>
</body>
```

Node - 부모

```
> const red = document.getElementById('red');
< undefined
> red.parentNode
<   ▶<ul id="color">…</ul>
> red.parentElement;
<   ▶<ul id="color">…</ul>
```

```
> document.documentElement.parentNode
<   ▶#document
> document.documentElement.parentElement
< null
```

Node – 자식(1)

```
> const ul = document.getElementById('color')
< undefined
> ul.childNodes   // 자식 노드 전부
< ▶ NodeList(9) [text, li#red, text, comment, text, li#blue, text, li#green, text]
> ul.children   // 자식 노드 중 요소 노드만 반환
< ▶ HTMLCollection(3) [li#red, li#blue, li#green, red: li#red, blue: li#blue, green: li#green]
```

★ NodeList임에도 불구하고 실시간 업데이트

```
> ul.childNodes   // 자식 노드 전부
< NodeList(9) [text, li#red, text, comment, text, li#blue, text, li#green,
  text]
    0: text
    1: li#red
    2: text
    3: comment
    4: text
    5: li#blue
    6: text
    7: li#green
    8: text
    length: 9
  ▶ [[Prototype]]: NodeList
```

text ⇥ 공백, 줄바꿈 포함

```
<ul id="color">
    <li id="red">red</li>    <!-- 0: text(줄바꿈), 1: li#red -->
    <!-- 노드 연습 중 -->    <!-- 2: text(줄바꿈), 3: commnet -->
    <li id="blue">blue</li>    <!-- 4: text(줄바꿈), 5: li#blue -->
    <li id="green">green</li>    <!-- 6: text(줄바꿈), 7: li#green -->
</ul>    <!-- 8: text(줄바꿈) -->
```

Node – 자식(2)

```
> ul.firstChild   // ul의 첫 번째 노드
<· ▶ #text
> ul.lastChild   // ul의 마지막 노드
<· ▶ #text
> ul.firstElementChild   // ul의 첫 번째 요소
<· ▶<li id="red">…</li>
> ul.lastElementChild   // ul의 마지막 요소 노드
<· ▶<li id="green">…</li>
```

Node – 형제

1.

```
> const blue = document.getElementById('blue')
< undefined
```

2. 형제 노드 중 모든 타입에서 가져오기

```
> blue.previousSibling
< ▶ #text
> blue.nextSibling
< ▶ #text
```

3. 원하는 결과를 얻으려면

```
> blue.previousElementSibling
< ▶ <li id="red">…</li>
> blue.nextElementSibling
< ▶ <li id="green">…</li>
```

Node

	모든 노드	요소 노드만
부모	parentNode	parentElement
자식	childNodes firstChild lastChild	children firstElementChild lastElementChild
형제	previousSibling nextSibling	previousElementSibling nextElementSibling

과제