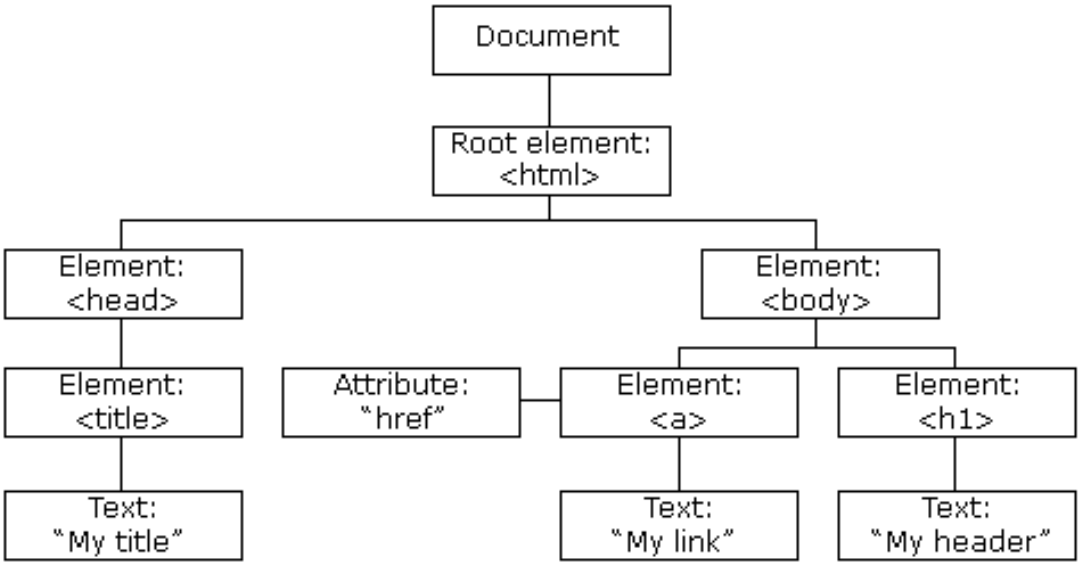
- 4주차 : JavaScript와 DOM -

/* DOM이란? */

- DOM(Document Object Model)은 XML, HTML 문서에 접근할 수 있는 문서 객체입니다.
- DOM 아래와 같은 tree 형식의 자료 구조를 갖고 있다.



- 자바스크립트는 새로운 HTML 요소나 속성을 추가할 수 있습니다.
- 자바스크립트는 존재하는 HTML 요소나 속성을 제거할 수 있습니다.
- 자바스크립트는 HTML 문서의 모든 HTML 요소를 변경할 수 있습니다.
- 자바스크립트는 HTML 문서의 모든 HTML 속성을 변경할 수 있습니다.
- 자바스크립트는 HTML 문서의 모든 CSS 스타일을 변경할 수 있습니다.
- 자바스크립트는 HTML 문서에 새로운 HTML 이벤트를 추가할 수 있습니다.
- 자바스크립트는 HTML 문서의 모든 HTML 이벤트에 반응할 수 있습니다.

출처 : http://tcpschool.com/javascript/js_dom_concept
출처 : https://www.w3schools.com/js/js_htmldom.asp

- 참고자료

- DOM의 구조 : https://youtu.be/B2UPRofY5ig
- Node : https://developer.mozilla.org/en-US/docs/Web/API/Node
- Element : https://developer.mozilla.org/en-US/docs/Web/API/Element
- Web API Interfaces : https://developer.mozilla.org/en-US/docs/Web/API

/* TagName, Id, ClassName */

- getElementsByTagName : 태그를 선택하는 메서드
- getElementById : ID를 선택하는 메서드
- getElementsByClassName : Class를 선택하는 메서드

```
var header = document.getElementsByTagName("header");
var serviceSection = document.getElementById("services");
var serviceSectionContainer = document.getElementsByClassName("container");

console.log(header);
console.log(serviceSection);
console.log(serviceSectionContainer);
```

- TagName, ClassName : 배열과 유사한 HTMLCollection 안에 복수의 값이 저장

```
var serviceSection = document.getElementById("services");
var serviceSectionContainer =
    serviceSection.getElementsByClassName("container");
```

/* querySelector(), querySelectorAll() */

```
var header = document.querySelector("header");
var serviceSection = document.querySelector("#services");
var serviceSectionContainer = document.querySelector(".container");
```

- querySelector : 가장 먼저 나오는 영역의 값을 가져온다.
- querySelectorAll : 복수의 영역의 값을 가져온다.

```
var serviceSectionContainerOne = document.querySelector("#services .container");
var serviceSectionContainerTwo = document.querySelectorAll(".container");

console.log(serviceSectionContainerOne);
console.log(serviceSectionContainerTwo[2]);
```

/* querySelectorAll() 응용 */

```
var serviceSectionContainer = document.querySelectorAll(".container");

for(var i = 0; i < serviceSectionContainer.length; i++){
    console.log(serviceSectionContainer[i]);
}
```

/* innerHTML, outerHTML */

- innerHTML : 자기 자신을 **제외**한 HTML 내용물을 String으로 반환
- outerHTML : 자기 자신을 **포함**한 HTML 내용물을 String으로 반환

```
var header = document.querySelector(".masthead");

console.log(header.innerHTML);
console.log(header.outerHTML);

console.log(typeof header.innerHTML);
console.log(typeof header.outerHTML);
```

/* textContent */

- 콘텐츠 정보를 가져오거나 이를 변경할 때 사용

```
var introHeading = document.querySelector(".intro-heading ");
console.log(introHeading.textContent);

introHeading.textContent = "Hello World";
```

- textContent와 innerHTML 차이점 : 태그 적용 가능여부

```
var introHeading = document.querySelector(".intro-heading ");

introHeading.textContent = "<em>Hello</em> World";
introHeading.innerHTML = "<em>Hello</em> World";
```

/* innerText */

- 콘텐츠 정보를 가져오거나 이를 변경할 때 사용

```
var introHeading = document.querySelector(".intro-heading ");

console.log(introHeading.innerText);
```

- innerText와 textContent 차이점 : CSS 적용 상태를 포함하는지 유무

```
var introHeading = document.querySelector(".intro-heading ");

console.log(introHeading.innerText);
console.log(introHeading.textContent);
```

/* 콘텐츠를 추가하는 메서드 */

- **createElement()** : 새로운 태그를 생성할 때 사용

```
// 태그 생성
var h3Test = document.createElement("h3");

// 태그 안에 콘텐츠 작성
h3Test.textContent = "Hello World";
console.log(h3Test);
```

- **appendChild()** : 생성된 태그를 부모 요소의 마지막 자식으로 추가

```
var h3Test = document.createElement("h3");
h3Test.textContent = "Hello World";

var introText = document.querySelector(".intro-text");
introText.appendChild(h3Test);
```

- appendChild 사용시 주의점
- 형제 영역에 appendChild()를 각각 사용할 경우 마지막 형제에게만 정상적으로 적용된다.

```
var introLeadIn = document.querySelector(".intro-lead-in");
var introHeading = document.querySelector(".intro-heading");
var introBtn = document.querySelector(".intro-text .btn");

var h3Txt = document.createElement("h3");
h3Txt.textContent = "?!?!?!";

introLeadIn.appendChild(h3Txt);
introHeading.appendChild(h3Txt);
introBtn.appendChild(h3Txt);
```

- **insertAdjacentHTML()** : 원하는 위치에 태그를 배치하고자 할 때 사용

```
<!-- beforebegin -->
<p>
    <!-- afterbegin -->
    content
    <!-- beforeend -->
</p>
<!-- afterend -->
```

```
var introHeading = document.querySelector(".intro-heading");
var txt = '<h2>Skydiving is fun!</h2>';

// beforebegin, afterbegin, beforeend, afterend
introHeading.insertAdjacentHTML('afterbegin', txt);
```

/* 콘텐츠를 제거하는 메서드 */

- removeChild() : 부모 안에 있는 내용물을 제거할 때 사용

```
var introText = document.querySelector(".intro-text");
var introHeading = document.querySelector(".intro-heading");

introText.removeChild(introHeading);
```

- 하지만 자기 자신을 제거하기 위해서는 무조건 부모를 거쳐야 함

```
var introText = document.querySelector(".intro-text");

introText.parentElement.removeChild(introText);
```

- remove() : 부모를 경유하지 않고 다이렉트로 해당 태그를 제거할 때 사용

```
var introText = document.querySelector(".intro-text");

introText.remove();
```

/* firstElementChild : 첫 번째 자식에 접근 */

```
var introText = document.querySelector(".intro-text");

console.log(introText.firstElementChild);
```

/* CSS 속성을 적용하는 방법 */

```
var introHeading = document.querySelector(".intro-heading");

introHeading.style.color = 'red';
introHeading.style.backgroundColor = 'pink';
introHeading.style.fontSize = '25px';
```

- cssText 속성을 사용하여 한 줄로 정리

```
var introHeading = document.querySelector(".intro-heading");

introHeading.style.cssText = "color: red; background-color: pink; font-size: 25px;";
```

- setAttribute 속성을 사용하여 태그에 직접 style 속성을 추가

```
var introHeading = document.querySelector(".intro-heading");

introHeading.setAttribute("style", "color: red; background-color: pink; font-size:
25px;");
```

- id 설정 후 스타일 적용

```
var introLeadIn = document.querySelector(".intro-lead-in");
introLeadIn.nextElementSibling.setAttribute("id", "heading-test");

// document.querySelector("#heading-test").style.backgroundColor = "pink";
// introLeadIn.nextElementSibling.style.backgroundColor = "pink";
```

/* Class 접근 방법 */

- **className** : 클래스 속성값을 가져오거나 클래스를 변경할 때 사용

```
var introHeading = document.querySelector(".intro-heading");
var introheadingclass = introHeading.className;
const arrayOfClasses = introheadingclass.split(' ');

// console.log(introheadingclass);
// console.log(arrayOfClasses);
```

- className 프로퍼티로 기존 클래스를 변경할 수 있다.

```
var introHeading = document.querySelector(".intro-heading");
introHeading.className = "test";
```

- **classList** : 복수의 클래스를 배열과 유사한 형태로 가져올 때 사용 (DOMTokenList 형태로 반환)

```
var introHeading = document.querySelector(".intro-heading");
var introheadingclass = introHeading.classList;

console.log(introheadingclass);
```

- **add(), remove(), toggle(), contains()**

```
var introHeading = document.querySelector(".intro-heading");
var introheadingclass = introHeading.classList;

introheadingclass.add("test");                                  // 클래스 추가
introheadingclass.remove("text-uppercase");                     // 클래스 삭제
introheadingclass.toggle("abc");                                // 클래스 추가, 삭제
console.log(introheadingclass.contains("intro-heading"));       // 클래스 확인
```

/* Event : https://developer.mozilla.org/en-US/docs/Web/Events */

- **addEventListener**() : 이벤트를 추가하는 함수

```
var introHeading = document.querySelector(".intro-heading");

introHeading.addEventListener('click', function () {
    console.log('Click!!!');
});
```

- **removeEventListener**() : 이벤트를 제거하는 함수

```
var introHeading = document.querySelector(".intro-heading");

function clickFunc() { console.log('Click'); }

introHeading.addEventListener('click', clickFunc);
introHeading.removeEventListener('click', clickFunc);
```

- 단, 아래와 같은 방식은 적용되지 않음 (원시 타입과 참조 타입 차이점)

```
var introHeading = document.querySelector(".intro-heading");

introHeading.addEventListener('click', function clickFunc() {
    console.log('Click');
});

introHeading.removeEventListener('click', function clickFunc() {
    console.log('Click');
});
```

- preventDefault() : 태그가 갖고 있는 기본 기능을 막을 때 사용

```
var links = document.querySelectorAll('.navbar-collapse a');
var serviceBtn = links[0];

serviceBtn.addEventListener("click", function(e) {
    e.preventDefault();
    console.log("Hello World");
});
```

/* API */

- axios : https://github.com/axios/axios
- HTTP 통신 라이브러리로 아직 베타버전이지만 깃허브에서 55,000개가 넘은 스타를 받을 정도로 가장 핫한 라이브러리 중 하나이다.

- NY API : https://developer.nytimes.com/

```
var api = “NY API키 삽입";

var cityStr;
var nytElem = document.getElementById("nytimes-articles");

var form = document.getElementById(“form-container");

form.addEventListener("submit", function (e) {

    e.preventDefault();
    cityStr = document.getElementById("city").value;

});
```

```
(function () {

    ……
    form.addEventListener("submit", function (e) {


        e.preventDefault();
        cityStr = document.getElementById("city").value;
        var nytimesUrl = `https://api.nytimes.com/svc/search/v2/articlesearch.json?q=${cityStr}&sort=newest&api-key=${api}`;


        fetch(nytimesUrl)
            .then(response => response.json())
            .then(data => {
                console.log(data.response.docs);
                var data = data.response.docs;
                data.forEach(function (item) {
                    nytElem.insertAdjacentHTML('beforeend', `
                        <li class="article">
                            <a href="${item.web_url}">${item.headline.main}</a>
                            <p>${item.snippet}</p>
                        </li>

                    `);
                })
            })
            .catch(err => console.error(err));


        document.getElementById("city").value = "";

    });

})();
```

```
form.addEventListener("submit", function (e) {

    ……

    var nytimesUrl = `https://api.nytimes.com/svc/search/v2/articlesearch.json?q=${cityStr}&sort=newest&api-key=${api}`;

    axios.get(nytimesUrl)
        .then(function (response) {

            var data = response.data.response.docs;
            // console.log(data);

            data.forEach(function (item) {

                nytElem.insertAdjacentHTML('beforeend', `
                    <li class="article">
                        <a href="${item.web_url}">${item.headline.main}</a>
                        <p>${item.snippet}</p>
                    </li>

                `);

            })

        })
        .catch(function (err) {
            console.log(err);
        })

});
```