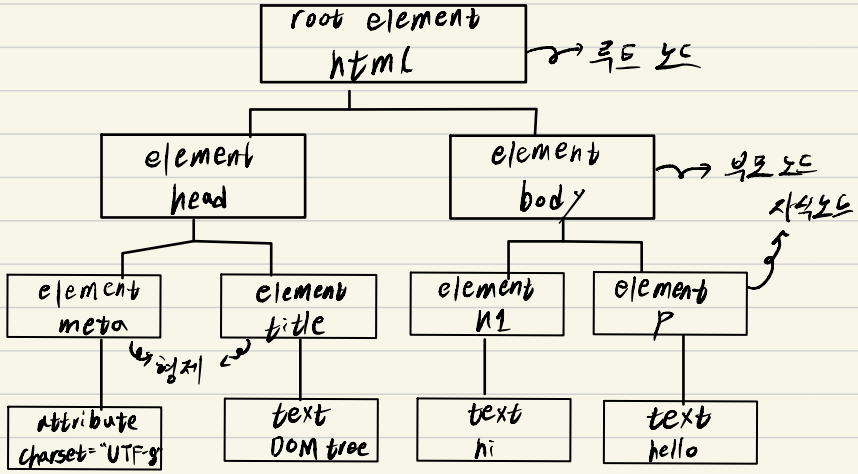
DOM

DOM(Document Object Model) : JS를 이용하여 웹 문서에 접근/제어 할 수 있도록 객체를 사용하여 웹 문서를 계층적으로 정의하는 방법

DOM 트리



node : 가지가 갈라져 나간 항목

DOM에 접근하기

method : getElementById()

↳ 특정한 ID가 포함된 요소를 반환함

ex) document.getElementById("heading");

⇒ Id 값이 heading인 요소를 반환

method: getElementsByClassName()

↳ 지정한 class 선택자 이름이 들어있는 요소들을 HTMLcollection 객체에 저장하여 반환

↳ 배열과 비슷하나, 배열이 아님

Class 속성 값이 같은 요소가 여러개일 수 있으므로 Elements

ex) document.getElementsByClassName("heading");

method: getElementsByTagName();

↳ 지정한 태그명을 사용한 요소들을 HTMLcollection 객체에 저장하여 반환

위의 두 method는 HTMLcollection을 반환한다.

↓

요소 노드만 저장, 텍스트 노드/속성 노드 등은 저장 안됨

↙ 해결방법

method: querySelector(), querySelectorAll()

↓ 반환 값이 하나인 경우 (id 등)

↳ 반환 값이 여러개인 경우 (class, 태그 이름 등)

↳ Node list 사용

ex) document.querySelector("#heading");

document.querySelectorAll(".bright");

property: innerText, innerHTML

↳ 원 요소의 내용을 수정하는 property. innerText는 텍스트 내용만, innerHTML은 태그까지 수정 가능하다.

ex) document.getElementById("current").innerText = "hi";

document.getElementById("current").innerHTML = "" + "hi" + ""

method : getAttribute()

↳ 해당 요소의 속성 값을 가져올 수 있는 메서드

ex) document.querySelector("#cup").getAttribute("src");

⇒ id가 cup인 요소의 src 속성을 반환함

method : setAttribute()

↳ 해당 요소의 속성 값을 지정

ex) document.querySelector("#cup").setAttribute("src", "image.jpg");

⇒ id가 cup인 요소의 src 속성 값을 image.jpg로 지정

DOM

이벤트처리

DOM 요소에 함수 직접 연결

```
ex) let cup = document.querySelector("#cup");
    cup.onclick = function() { ... }
```

함수 이름을 사용해 연결

```
ex) let cup = document.querySelector("#cup");
    cup.onclick = changePic; // 함수 뒤에 괄호를 붙이지 않음 ☆

    function changePic() { ... }
```

이벤트 객체 : DOM에는 이벤트의 정보를 저장하는 event 객체가 있다.

※ event의 '정보'를 저장할 뿐, 이벤트가 발생한 대상을 저장하진 않는다.

```
ex) let cup = document.querySelector("#cup");
    cup.onclick = function(event) {
        alert("이벤트 유형 : " + event.type);
    }
```

→ 이벤트 발생시, 이벤트 처리기에 이벤트 객체가 인수로 넘어옴

프로퍼티	ctrlKey	이벤트가 발생했을 때 Ctrl 를 눌렀는지 여부를 boolean값으로 반환합니다.
	pageX	현재 문서 기준으로 이벤트가 발생한 가로 위치를 반환합니다.
	pageY	현재 문서 기준으로 이벤트가 발생한 세로 위치를 반환합니다.
	screenX	현재 화면 기준으로 이벤트가 발생한 가로 위치를 반환합니다.
	screenY	현재 화면 기준으로 이벤트가 발생한 세로 위치를 반환합니다.
	shiftKey	이벤트가 발생할 때 Shift 를 눌렀는지 여부를 boolean값으로 반환합니다.
	target	이벤트가 최초로 발생한 대상을 반환합니다.
	timeStamp	이벤트가 발생한 시간을 반환합니다.
	type	발생한 이벤트 이름을 반환합니다.
	which	키보드와 관련된 이벤트가 발생할 때 키의 유니코드값을 반환합니다.
메서드	preventDefault()	이벤트를 취소할 수 있는 경우에 취소합니다.

이벤트가 발생한 요소

event 객체는 이벤트 정보만 포함하기 때문에, 발생한 대상에 접근하려면 예약어 this를 사용

```
ex) let card = document.querySelector("#card");
    card.onclick = function (event) {
        alert("클릭한 파일:" + this.src); //id가 card인 요소의 src 속성값
    }
```

method : addEventListener()

↳ 한 요소에 여러 이벤트 처리기를 연결할 수 있음

기본형 : 요소.addEventListener(이벤트, 함수, 캡처 여부);

이벤트 → on을 붙이지 않음

캡처 여부 → 기본 값은 false

```
ex) card.addEventListener("mouseover", changepic);
```

↳ 괄호는 없음

CSS속성 접근하기

JS를 이용하여 스타일 속성 값을 변경할 수 있다.

기본형: document.요소명.style.속성명 = 속성 값;

ex) document.getElementById("#desc").style.color = "blue";

DOM에서 노드 추가·삭제

노드리스트 : 여러개의 노드들의 정보를 저장하고 있는 객체

노드 추가

노드 추가의 과정 ⇒

Text 노드를 포함하는 예시

1. 요소 노드를 만든다
2. 텍스트 노드를 만든다.
3. 요소 노드를 부모, 텍스트 노드를 자식으로 연결한다.
4. 웹 문서 내의 요소 노드를 부모, 새로 만든 요소 노드를 자식으로 연결한다

method : createElement()

↳ 새로운 요소 노드를 만든다.

ex) let newP = document.createElement("p"); --- ①

method : createTextNode()

↳ 새로운 텍스트 노드를 만든다.

ex) let newText = document.createTextNode("hi"); --- ②

method : appendChild()

↳ 부모 노드와 자식 노드를 연결한다. 연결된 자식은 맨 끝에 추가된다.

기본형: 부모노드.appendChild(자식노드);

ex) newP.appendChild(newText); --- ③

document.getElementById("Info").appendChild(newP);

--- ④

※ 속성 값이 있는 새로운 노드 추가

↳ method : createElement() // 새 요소 제작
method : createAttribute() // 새 속성 노드 제작
property : value // 새 속성 노드의 속성값 설정
method : setAttributeNode() // 새 요소와 속성노드 연결
method : appendChild() // 현 문서의 요소와 새 요소 연결

```
ex) let newImg = document.createElement("img");
    let srcNode = document.createAttribute("src");
    srcNode.value = "dom.jpg";

    newImg.setAttributeNode(srcNode);
    document.getElementById("Info").appendChild(newImg);
```

노드 삭제

노드 삭제의 과정 ⇒ 1. 삭제해야 할 노드의 부모 노드를 찾는다.
2. 부모 노드를 기준으로 삭제할 노드(자식)를 삭제한다.

property : parentNode

↳ 현재 노드의 부모 노드를 (요소) 반환함
기본형 : 노드.parentNode

method : removeChild()

↳ 자식 노드를 삭제함
기본형 : 부모노드.removeChild(자식 노드);

```
ex) li.parentNode.removeChild(li);
```

⇒ li 노드 삭제