DOM의 기초1

참고 URL

https://www.w3schools.com/jsref/dom_obj_document.asp

1) id로 접근

getElementById() 메서드는 지정된 값을 갖는 id 속성의 Element Object를 반환합니다.

```
document.getElementById(elementID)

Return Value : An Element Object
```

Element Object는 Node object의 다른 유형이라고 할 수 있습니다. 단지 텍스트 노드와 주석과 같은 것들은 Element Object에 포함되지 않습니다.

활용 구문입니다.

```
var tutorial=document.getElementById("tutorial");
tutorial.style.backgroundColor="#666";


style은 Element Object의 스타일 리스트를 얻거나 작성할 수 있습니다.

참고 URL
https://www.w3schools.com/jsref/prop_html_style.asp
```

2) tag 이름으로 접근

getElementsByTagName() 메서드는 지정된 태그 이름을 가진 문서의 모든 요소를 NodeList object로 반환합니다.

```
document.getElementsByTagName(tagname)

Return Value : A NodeList object
```

NodeList object는 단순한 Array과 같은 노드 목록입니다.

활용 구문입니다.

```
var li=document.getElementsByTagName("li");
// console.log(li);
// console.log(li[1]);
// console.log(li.item(1));
// console.log("li length : "+li.length);

for(var i=0; i<li.length; i++){
    li[i].style.backgroundColor="#666";
    // console.log("background color : "+li[i].style.backgroundColor);
}
```

li로 참조된 NodeList object는 배열처럼 배열자를 참조할 수 있습니다.

3) class 이름으로 접근

getElementsByClassName() 메서드는 지정된 클래스 이름을 가진 문서의 모든 요소를 NodeList object로 반환합니다.

```
document.getElementsByClassName(classname)

Return Value : A NodeList object
```

활용 구문입니다.

```
var html=document.getElementsByClassName("html");
// console.log(html);
// console.log(html[0]);
// console.log(html.item(0));
html[0].style.backgroundColor="#666";
```

4) CSS 기법으로 첫 번째 노드 선택

querySelector() 메서드는 문서의 지정된 CSS 스타일과 일치하는 첫 번째 Node object로 반환합니다.

```
document.querySelector(CSS selectors)

Return Value : A Node object
```

활용 구문입니다.

```
var li=document.querySelector("#tutorial > li");
li.style.backgroundColor="#666";
```

5) CSS 기법으로 전체 노드 선택

querySelectorAll() 메서드는 지정된 CSS 스타일과 일치하는 문서의 모든 요소를 정적 NodeList object로 반환합니다.

```
document.querySelectorAll(CSS selectors)

Return Value : A NodeList object
```

활용 구문입니다.

```
var li=document.querySelectorAll("#tutorial > li");

for(var i=0; i<li.length; i++){
    // console.log(li[i]);
    // console.log("text content : "+li[i].textContent);

    if(li[i].textContent == "HTML" || li[i].textContent == "CSS"){
        li[i].style.backgroundColor="#666";
    }
}


textContent 속성은 Node object의 텍스트 내용을 얻거나 작성할 수 있습니다.
```

참고 URL

https://www.w3schools.com/jsref/prop_node_textcontent.asp

6) 상위 노드에 접근

parentNode 속성은 지정된 노드의 부모 노드를 Node object로 반환합니다.

```
node.parentNode

Return Value : A Node object
```

활용 구문입니다.

```
var html=document.querySelector(".html");
var contUl=html.parentNode;
// console.log(contUl);
```

7) 상위 HTML 노드에 접근

parentElement 속성은 지정된 요소의 Element object를 반환합니다.

```
node.parentElement

Return Value : An Element object
```

활용 구문입니다.

```
var html=document.querySelector(".html");
var contUl=html.parentElement;
// console.log(contUl);
```

8) 하위 노드에 접근

childNodes 속성은 노드의 자식 노드 컬렉션을 NodeList object로 반환합니다.

```
element.childNodes

Return Value : A NodeList object
```

활용 구문입니다.

```
var cont=document.querySelector("#tutorial");
var contLi=cont.childNodes;
// console.log(contLi);
```

9) 하위 HTML 노드에 접근

children 속성은 요소의 자식 요소 컬렉션을 HTMLCollection object로 반환합니다.

```
element.children

Return Value : A live HTMLCollection object
```

NodeList object는 모든 Node object 유형을 포함할 수 있지만, HTMLCollection은 Element Object만 포함됩니다. NodeList object와 HTMLCollection object는 동일한 메서드를 제공합니다.

활용 구문입니다.

```
var cont=document.querySelector("#tutorial");
var contLi=cont.children;
// console.log(contLi);
// console.log(contLi[0]);

for(var i=0; i<contLi.length; i++){
    contLi[i].style.backgroundColor="#666";
}
```

10) 하위 처음 노드에 접근

firstChild 속성은 지정된 노드의 첫 번째 자식 노드를 Node object로 반환합니다.

```
node.firstChild

Return Value : A Node object
```

활용 구문입니다.

```
var cont=document.querySelector("#tutorial");
var contChild=cont.firtChild;
// console.log(contChild);
```

11) 하위 처음 HTML 노드에 접근

firstElementChild 속성은 지정된 요소의 첫 번째 자식 요소를 반환합니다.

```
node.firstChild

Return Value : A Node object
```

활용 구문입니다.

```
var cont=document.querySelector("#tutorial");
var contChild=cont.firstElementChild;
// console.log(contChild);
```

12) 하위 마지막 노드에 접근

lastChild 속성은 지정된 노드의 마지막 자식 노드를 Node object로 반환합니다.

```
node.lastChild

Return Value : A Node object
```

활용 구문입니다.

```
var cont=document.querySelector("#tutorial");
var contChild=cont.lastChild;
// console.log(contChild);
```

13) 하위 마지막 HTML 노드에 접근

lastElementChild 속성은 지정된 요소의 마지막 자식 요소를 반환합니다.

```
node.lastElementChild

Return Value : A Node object
```

활용 구문입니다.

```
var cont=document.querySelector("#tutorial");
var contChild=cont.lastElementChild;
// console.log(contChild);
```

14) 이전 노드에 접근

previousSibling 속성은 동일한 트리 수준에서 지정된 노드의 이전 노드를 반환합니다.

```
node.previousSibling

Return Value : A Node object
```

활용 구문입니다.

```
var css=document.querySelector(".css");
var previous=css.previousSibling;
// console.log(previous);
```

15) 이전 HTML 노드에 접근

previousElementSibling 속성은 동일한 트리 수준에서 지정된 요소의 이전 요소를 반환합니다.

```
node.previousElementSibling

Return Value : A Node object
```

활용 구문입니다.

```
var css=document.querySelector(".css");
var previous=css.previousElementSibling;
// console.log(next);
previous.style.backgroundColor="#666";
```

16) 다음 노드에 접근

nextSibling 속성은 동일한 트리 수준에서 지정된 노드 바로 다음 노드를 반환합니다.

```
node.nextSibling

Return Value : A Node object
```

활용 구문입니다.

```
var html=document.querySelector(".html");
var next=html.nextSibling;
// console.log(next);
```

17) 다음 HTML 노드에 접근

nextElementSibling 속성은 지정된 요소 바로 다음의 요소를 같은 트리 수준에서 반환합니다.

```
node.nextElementSibling

Return Value : A Node object
```

활용 구문입니다.

```
var html=document.querySelector(".html");
var next=html.nextElementSibling;
// console.log(next);
next.style.backgroundColor="#666";
```