**Document 객체**

Document Object model 줄여서 dom이라고 읽는다. 이 문서 객체 모델은 html문서의 구조를 나타내는 표기법인 동시에 브라우저가 html문서를 조작하는 핵심이다.

Dom은 트리 구조로 표현 하는데, 이 트리는 노드 라는 자료구조로 구성됩니다. 루트노드를 제외하면 모든 노드에 부모가 있으며, 자식 도느는 있어도 되고 없어도 됩니다. 루트 노드는 document 객체이며 자식 노드는 html요소 하나 뿐입니다.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>Document</title>

</head>

<body>

</body>

</html>

( 이 html을 트리로 그려주자 )

Dom트리의 모든 노드는 node클래스의 인스턴스이다.

트리는 매우 단순하고 직관적인 개념이다. 부모노드는 직접적인 노드의 부모입니다. 다시 말해 어떤 노드(태그)의 바로 윗 단계를 말합니다. 자식 노드 역시 직접적인 자식입니다. 자손은 자식, 자식의 자식등을 말합니다. 조상은 부모 , 부모의 부모등을 말합니다.

**Get메서드**

Dom에는 원하는 html요소를 빨리 찾을 수 있는 메서드가 있습니다. 가장 먼저 볼 것은 document.getElementById입니다. 모든 html요소는 고유한 id를 할당 받을 수 있습니다. 이 때 이 메서드는 자손노드의 Id프로퍼티를 찾아 요소를 반환합니다.

(id는 html내부에서 유일한 값이 되어야만 하지만 여러 요소에 같은 id값을 주는 것이 가능하다. 그러나 이 때 해당 메서드는 document객체의 자손들을 뒤져보며 찾는 id와 일치하고 처음 만나는 것을 반환한다. 그러니 꼭 이런 규칙들을 지키면서 코딩하자)

다음 메서드는 getElementsByClassName()입니다. 이 클래스라는 녀석은 같은 성격의 그룹을 말합니다. 그 말인 즉 여러개의 요소들을 반환 할 겁니다. 그렇다면 여러 요소들을 담을 수 있는 데이터타입은 무엇이 있을 까요 ? 예 맞습니다. 배열입니다. 이 메서드는 배열을 반환합니다.

다음 메서드는 getElementByTagName입니다. 이 메서드는 무엇을 반환하는지 감이 오시죠 ?

네 해당 태그명을 가진 요소들을 담은 배열을 반환하게 됩니다.

**쿼리**

방금 가르쳐 드렸던 3가지의 get메서드는 id,클래스,태그명 같은 단일 조건으로 요소를 지정하기에 제한이 많습니다. 그러나 쿼리메서드들은 한가지 조건이 아니라 다른 요소와의 관계를 사용해 원하는 요소를 찾는 훨씬 더 강력하고 범용적인 메서드들입니다. 바로 css선택자를 사용해 요소를 찾는 메서드 입니다.

Document.querySelector()는 인자로 스트링으로 작성된 선택자를 받습니다.

예제를 해볼까요 ?

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>Document</title>

<style>

#first{

background-color: black;

}

.firstClass{

width: 100px;

height: 100px;

}

.secondClass{

background-color: red;

}

.blue{

width: 15px;

height: 15px;

background-color: blue;

}

</style>

<script>

console.log(Node.prototype);

</script>

</head>

<body>

<div id="first" class="firstClass"></div>

<div class="firstClass secondClass"></div>

<div class="firstClass secondClass"></div>

<div class="firstClass secondClass">

<div class="blue">

</div>

</div>

</body>

<script>

console.log(document.querySelector("#first.firstClass"));

console.log(document.querySelector(".firstClass.secondClass"));

console.log(document.querySelector(".firstClass.secondClass:nth-child(3)"));

console.log(document.querySelector(".firstClass .blue"));

</script>

</html>

두번째는 document.querySelectAll입니다. 이 메서드는 해당 선택자를 가진 요소를 가진 배열을 반환합니다. 예제를 만들어 보자

1. 선택자를 이용해서 firstClass의 firstId라는 아이디를 자식 가진 요소를 선택

**이것으로 무엇을 할 수 있나 ?**

콘텐츠 먼저 수정 해봅시다. 모든 요소에는 textContent와 innerHTML이라는 프로퍼티가 있으며 이 프로퍼티를 통해 요소의 콘텐츠에 접근 하거나 수정 할 수 있다.

textContent메서드는 값으로 할당한 html요소를 모두 제거하고 순수한 텍스만 제거하고 innerHTML을 html태그를 그대로 제공합니다.

<body>

<div id="textContent">

</div>

<div id="innerHTML">

</div>

</body>

<script>

let innerHTML = document.getElementById("innerHTML");

let textContent = document.getElementById("textContent");

innerHTML.innerHTML = "<h1>hello innerHTML</h1>";

textContent.textContent = "<h1>hello innerHTML</h1>";

</script>

Dom요소 만들기

innerHTML 프로퍼티를 사용해서 dom노드를 새로 만들 수 있다는 것은 보았다. 그러나 document.createElement메서드를 사용하여 새 노드를 만들수도 있다. 이 함수는 새로운 요소를 만들지만 dom에 추가하지는 않는다. Dom트리에 이 메서드로 만든 요소를 넣으려면 어떤 노드의 어디에 넣어 줄지 결정해야한다.

let pTag1 = document.createElement('p')

let pTag2 = document.createElement('p')

// 메서드의 인자로 스트링으로 작성된 태그명을 전달한다.

pTag1.innerText = "hello world";

pTag2.innerText = "i'm lee jong ho";

const parent = document.getElementById("parent");

parent.appendChild(pTag1);

parent.appendChild(pTag2);

//자식요소 중 마지막에 넣는 메서드

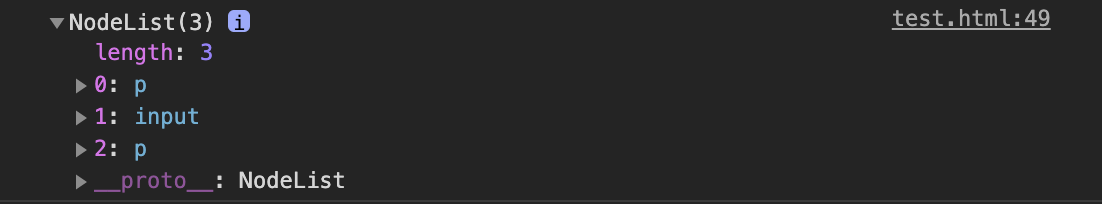
let input = document.createElement('input');

parent.insertBefore(input,pTag2)

//2번째 인자의 요소보다 앞에 새로운 dom node를 추가하는 메서드

appendChild나 insertBefore를 보니 관리하고 있는 프로퍼티 배열이 있는 듯하다.

console.log(parent.childNodes);



**엘리먼트 스타일링**

Html 에서 style을 지정하는 방법은 css를 이용하여, class id를 요소에 지정 하는 방법이 있고 html태그에서 직접 지정하는 inline방식이 있다. Inline방식을 사용하듯 엘리먼트 요소에 직접 style을 지정 하려면 엘리먼트 요소의 style 프로퍼티를 이용하여 직접 지정 하면된다 .

pTag1.style.color = "red";

pTag1.style.borderBottom = "1px solid red";

이렇게 하나하나 스타일 프로퍼티를 지정하기는 힘드니 요소를 대량으로 생산할 때는 클래스를 추가하는 것이 현명 할 것이다. Dom 노드에는 클래스를 추가 할 수 있는 메서드와 제거 할 수 있는 메서드가 제공 된다.

pTag1.classList.add('firstClass');

pTag1.classList.add('white');

pTag1.classList.remove('firstClass');

다른 방법은 className프로퍼티를 이용하는 방법인데, 이 프로퍼티는 스트링형태로 작성된다.

pTag1.className = 'firstClass white';

// pTag1.classList.add('firstClass');

// pTag1.classList.add('white');

**이벤트**

어떤 요소를 클릭하거나 키보드를 입력하여 특정 함수를 실행 시키는 것을 이벤트라고 한다. 이 이벤트 종류는 엄청나게 많아 전부 다뤄 보지 못하고 클릭이벤트와 키보드이벤트 정도만 다루어 보도록 하겠습니다.

Html에서 인라인 방식으로 이벤트를 설정하는 방법은 다음과 같습니다.

<div id="parent" onclick=clickEvent()>

클릭하면 경고창

</div>

</body>

<script>

// function clickEvent(){

// alert("클릭 이벤트");

// }

let clickEvent = function(){

alert("클릭이벤트")

}

</script>

이벤트를 발생시킬 요소에 이벤트 어트리뷰트에 실행할 함수 식별자와 호출자를 할당합니다. 이렇게 작성하면 해당 이벤트가 발생 했을 때 특정 함수를 호출합니다.

그렇지만 동적으로 createElement를 이용하여 만들었을 경우 저런 방식을 사용하지 못하므로 엘리먼트 요소에 이벤트를 할당하는 방법을 알아야 합니다 .

요소에 이벤트를 할당하는 방법은 addEventListener 메서드를 사용하는 방법입니다.

<body>

<button onclick=clickEvent()>요소 만들기 버튼</button>

<div id="parent">

</div>

</body>

<script>

let clickEvent = function(){

const parent = document.getElementById('parent');

const child = document.createElement('div');

child.innerText = "child";

child.classList.add('child');

child.addEventListener('click',event=>{

//addEventListener메서드는 첫번째 전달인자로 이벤트를 지정한다.

//두번째 전달인자는 이벤트가 일어났을 때 실행할

//한개의 매개변수를 가지는 콜백함수를 전달 받는다.

parent.removeChild(event.target);

//이 한개의 매개변수는 해당 이벤트에 관한 정보가 들어 있다.

//event.target이라는 프로퍼티는 이벤트가 일어나는 노드가 들어있다.

//clientX,ClientY같은 해당 이벤트가 일어난 좌표도 있다.

})

parent.appendChild(child);

}

</script>

이 이벤트 객체에는 이벤트를 막는 메서드도 있다. 무슨 뚱딴지 같은 소리인가 하시는 분들이 있을 것이다. 이 preventDefault()메서드의 목적은 a태그나 input type submit같은 기본적으로 장착된 이벤트들을 원하는 조건에 맞지 않다면 데이터 전송 혹은 링크 이동을 막기 위함이다.

간단하게 로그인 창을 예를 들어보겠다. 다음과 같이 구성되어있을 때 submit버튼을 누르면 form에 해당하는 내용들이 target으로 전달될 것이다. 그러나 아이디나 비밀번호를 적지 않았음에도 전송되기 때문에 해당 버튼을 눌러도 아이디와 비밀번호가 없다면 전송을 하면 안된다.

<div id="parent">

<form method="POST" target="http://local.com/login">

아이디 : <input id="idInput">

비밀번호 : <input id="pwInput">

<input type="submit">

</form>

</div>

이제 이벤트를 이용해서 전송을 막아보도록 하겠다.

const submitButton = document.getElementById('submit');

submitButton.addEventListener('click',function(event){

const idInput = document.getElementById('idInput').value;

const pwInput = document.getElementById('pwInput').value;

//input창 같은 텍스트 필드의 값들은 요소의 value프로퍼티에 쌓인다.

if(idInput.length < 4 || pwInput.length < 4){

event.preventDefault();

console.log(event);

alert("아이디와 비밀번호를 입력해주세요");

};

})

다음과 같이 만들어 조건을 무시하고, 클릭하면 폼의 제출이 제대로 되지 않는 것을 볼 수 있다.