39장 DOM -1-

39.0 머릿말

- DOM이란?
  - 브라우저 엔진에 인한해 파싱 HTML 문서를 파싱하여 브라우저가 이해할 수 있는 자료구조로 생성한 결과물
  - HTML 문서의 계층적 구조와 정보를 표현하며 이를 제어할 수 있는 API, 즉 프로그래밍 메서드를 제공하는 트리 자료구조

39.1 노드

1. HTML 요소와 노드 객체

- 하나의 태그 안에는 중첩관계를 가지게, 요소 노드, 어트리뷰트 노드, 텍스트 노드를 갖는다.
- 트리 자료구조
  - 계층적 구조를 표현하는 비선형 구조
    - 루트노드(최상위 노드), 리프노드(자식이 없는 노드), 부모노드, 자식노드, 형제노드(sibling)
    - 노드 객체들로 구성된 트리 자료구조를 DOM 이라고 한다. DOM트리라고도 부른다.

2. 노드 객체의 타입

- ** 공백 텍스트 노드
- 문서 노드
  - 루트 노드로 document 객체를 가지진다.
    - 전역객체 window의 document 프로퍼티
  - HTML 문서당 1개만 존재
  - DOM 트리 탐색의 노드들에 접근하기 위한 첫 진입점 역할
- 요소 노드
  - HTML 요소
  - 문서구조 표현
- 어트리뷰트 노드
  - 부모 노드와 연결 x
    - 요소 노드해만 연결
    - 요소 노드의 형제 노드 아니다.
  - 변경하려면 요소노드에 먼저 접근후 변경
- 텍스트 노드
  - 텍스트이며 요소노드의 자식 노드이다.
  - 어트리뷰트와 마찬가지로 요소노드에 먼저 접근해서 변경
  - 자식 노드을 가질 수 없는 리프노드
- 이외 총 12가지 종류가 있다.
  - 주석을 위한 Comment 노드
  - DOCTYPE을 위한 DocumentType 노드
  - 복수의 노드 생성하여 추가할 때 사용하는 DocumentFragment 노드

3. 노드 객체의 상속 구조

- DOM API : DOM을 구성하는 노드 객체는 자신의 구조와 정보를 제어할 수 있다.
  - 이를 통해 노드탐색, 노드조작, 노드생성이 가능해진다.
- DOM API 는 표준 빌트인 객체가 아니라 브라우저에서 제공하는 호스트 객체이다.
  - 하지만 노드 객체도 자바스크립트 객체이므로 프로토타입의 상속 구조를 갖는다.
- Object -> EventTarget -> Node -> (Document, Element, Attr, CharacterData) -> HTMLElement -> HTMLBodyElement...등 의 상속구조
  - 이러한 프로토타입 체인을 통한 상속 구조 덕분에 각 요소 노드 객체는 상위 체인에 있는 프로토타니 메서드를 상속받아 사용 할 수 있다.
  - 노드 객체는 공통된 기능일수록 프로토타입 체인의 상위에, 개별적인 고유 기능일수록 하위,에 배치를 구축하여 노드 객체에 필요한 기능을 제공

39.2 요소 노드 취득

0. 머릿말

- 요소 노드의 취득은 HTML 요소를 조작하는 시작점이다.

1. id를 이용한 요소 노드 취득

- Document.prototype.getElementById(' id값 ');
  - Document의 프로퍼티로 반드시 document 를 통해 호출해야함
- id 어트리뷰트 값을 갖는 하나의 요소 노드를 탐색하여 반환
- 값은 공 가 아니라면 첫 id만 반환한다.
- 만약 전달된 id값을 갖는 HTML의 요소가 존재하지 않는 경우 null을 반환한다.
- HTML 요소에 id 어트리뷰트를 부여하면 id값과 동일한 이름의 전역 변수가 암묵적으로 선언, 해당 노드 객체가 할당되는 부수 효과
  - 이러 이름이 같은 변수가 있다면 노드 객체 재할당 X

2. 태그 이름을 이용한 요소 노드 취득

- Document#/Element#getElementsByTagName(' 태그이름 ')
- Elements 가 복수형인 것에서 알 수 있듯이 DOM 컬렉션 객체인 HTMLCollection 객체 반환
  - 이는 유사배열이면서 이터러블
- HTML 의 모든 요소 노드 취득하려면 인수에 ' * ' 전달
- Document.prototype.getElementsByTagName 과 Element.prototype.getElementsByTagName 의 차이점
  - Document의 경우
    - DOM의 루트 노드인 문서 노드, 즉 document를 통해 호출하여 DOM전체에서 요소 노드를 탐색하여 반환
  - Element의 경우
    - 특정 요소 노드를 통해 호출하여, 특정 요소 노드의 자손 노드에서 요소 노드를 탐색하여 반환
- 태그 이름을 갖는 요소가 존재하지 않는 경우 빈 HTMLCollection 객체 반환

3. class를 이용한 요소 노드 취득

- Document#/Element#getElementsByClassName(' 클래스이름 ')
  - 공백으로 구분하여 여러개의 클래스 지정 가능
- 태그이름과 마찬가지로 HTMLCollection 반환
- 태그 이름과 마찬가지로 Document, Element 프로퍼티임으로 나뉜다.
- 클래스 이름을 갖는 요소가 존재하지 않는 경우 빈 HTMLCollection 객체 반환

4. CSS 선택자를 이용한 요소 노드 취득

- Document#/Element#.querySelector(' 선택자 ')
  - 인수로 전달된 CSS 선택자를 만족시키는 요소 노드가 여러 개인 경우 첫 번째 요소 노드만 반환
  - ex. 요소 노드가 존재 X => null 반환
  - CSS 선택자가 문법에 맞지 않는 경우 DOMException 에러 발생
- Document#/Element#.querySelectorAll(' 선택자 ')
  - 모든 요소 노드를 탐색하여 NodeList 객체 반환한다.
  - 인수에 전달된 요소가 존재 X => 빈 NodeList 객체 반환
  - 선택자 문법 X => DOMException 에러 발생
- 선택자 종류
  - * { ... } : 전체 선택자 (모든 요소를 선택)
  - p { ... }: 모든 p 태그 요소를 모두 선택
  - #foo : id 값
  - .foo : class 값
  - input[type=text] { ...} : 어트리뷰트 선택자 (input 요소 중 type 어트리뷰트 값이 'text'인 요소를 모두 선택)
  - div p : 후손 선택자(div 요소의 후손 요소 중 p 요소를 모두 선택)
  - div > p : 자식 선택자(div 요소의 자식 요소 중 p 요소를 모두 선택)
  - p + ul : 인접 형제 선택자 (p 요소의 형제 요소 중에 p 요소 바로 뒤에 위치하는 ul 요소 선택)
  - p ~ ul : 일반 형제 선택자(p 요소의 형제 요소 중에 p 요소 뒤에 위치하는 ul 요소를 모두 선택)
  - a:hover: 가상 클래스 선택자 (hover 상태인 a 요소를 모두 선택)
  - p::before : 가상 요소 선택자 (p요소의 콘텐츠의 앞에 위치하는 공간을 선택 -> 일반적으로 content 프로퍼티와 함께 사용)
- 태그 이름과 마찬가지로 Document, Element 프로퍼티임으로 나뉜다.
- getElementById, getElementsBy*** 메서드 보다 다소 느리다!!
  - 하지만 CSS 선택자 문법으로 좀 더 구체적인 조건으로 요소 노드를 취득할 수 있다는 장점이 있다!!!
- 권장 방식
  - Id 어트리뷰트가 있는 요소 노드를 취득할 땐 getElementById 를 사용
  - 그 외의 경우엔 querySelector, All 메서드를 사용하는 것을 권장

5. 특정 요소 노드를 취득할 수 있는지 확인

- Element.prototype.matches('css 선택자')
  - 인수로 전달된 CSS 선택자를 통해 특정 요소 노드를 취득할 수 있는지 확인
  - 불리언 값 반환
- 이벤트 위임을 사용할 때 유용하다.

6. HTMLCollection과 NodeList

- DOM 컬렉션 객체인 HTMLCollection 과 NodeList는 모두 유사 배열 객체이면서 이터러블이다.
  - for ...of 문으로 순회 가능하며 스프레드 문법을 사용해서 간단히 배열로 변환
- HTMLCollection 과 NodeList는 노드 객체에 상태 변화를 실시간으로 반영하는 살아 있는 객체이다.
  - HTMLCollection은 언제나 live 객체로 동작
  - NodeList는 대부분 실시간으로 반영하지 않고 과거의 정적 상태를 유지하지만, 경우에 따라 live 객체로 동작할 때도 있다.
- HTMLCollection
  - 항상 살아 있는 DOM 컬렉션 객체
  - getElementsByTagName
  - getElementsByClassName
  - for 문으로 태그의 class 어트리뷰트 변경시 실시간으로 HTMLCollection 객체에서 제거되는 사이드 이펙트때문에 생각대로 할 적용되지 않는 경우가 많기에 주의해서 써야 한다.
    - 이럴 땐 for 문을 역방향으로 순회하면 피할 수 있다.
    - while문을 사용하여 HTMLCollection 객체에 노드 객체가 남아 있지 않을 때까지 무한 반복하는 방법으로 회피 가능
    - 가장 좋은 방법은 HTMLCollection 객체를 사용하지 않는 것
    - 공이 써먹던다면 복사해서 [...Collection]과 배열 고차 함수를 사용하여 순회한다.
- NodeList
  - HTMLCollection 부작용을 해결하기 위해 querySelectorAll 메서드를 사용한다.
  - NodeList는 NodeList.prototype.forEach 메서드를 상속받아 사용한다.
    - 이 메서드는 실시간으로 노드 객체의 상태를 반영하지 않는 NodeList를 반환하기 때문
    - item, entries, keys, values 메서드 제공
  - childNodes 프로퍼티가 반환하는 NodeList 객체는 HTMLCollection 객체와 같이 실시간으로 노드 객체의 상태 변경을 반영
- HTMLCollection과 NodeList 객체는 예상과 다르게 동작할 때가 있어 다루기 까다롭다.
  - 노드 객체의 상태 변경과 상관없이 안전하게 사용하려면 두 객체를 배열로 변환해서 사용하는 것을 권장한다.
    - 스프레드 문법이나, Array.from 메서드를 사용하여 배열로 변환
  - 두 객체도 메서드를 제공하긴 하지만 배열 고차 함수만큼 다양하지 않다.

39.3 노드 탐색

0. 머릿말

- 요소 노드를 취득한 다음, 취득한 요소 노드를 기점으로 DOM 트리의 노드를 옮겨 다니며 노드들을 탐색해야 할 때가 있다.
- DOM 트리 상의 노드를 탐색할 수 있도록 Node, Element 인터페이스는 트리 탐색 프로퍼티를 제공한다.

- Node.prototype 제공 프로퍼티
  - parentNode
  - previousSibling
  - firstChild
  - childNodes
- Element.prototype 제공 프로퍼티
  - previousElementSibling
  - nextElementSibling
  - children
- 모든 접근자 프로퍼티는 setter없이 getter만 존재
  - 읽기 전용 접근자

1. 공백 텍스트 노드

- HTML 요소 사이의 스페이스, 탭, 줄바꿈(개행) 등의 공백 문자는 텍스트 노드를 생성
- 노드 탐색할 때 공백 텍스트 노드 주의해야 한다.
  - 이를 공백 텍스트 노드라고 함

2. 자식 노드 탐색

- Node.prototype.childNodes
  - 자식 노드를 모두 탐색 -> DOM 컬렉션 객체인 NodeList에 담아 반환
  - 요소 노드 뿐만 아니라 텍스트 노드도 포함되어 있을 수 있다.
- Element.prototype.children
  - 자식 노드 중 요소 노드만 모두 탐색 -> DOM 컬렉션 객체인 HTMLCollection 반환
  - 텍스트 노드 포함 X
- Node.prototype.firstChild
  - 첫 번째 자식 노드 반환
  - 텍스트 노드이거나 요소 노드
- Node.prototype.lastChild
  - 마지막 자식 노드 반환
  - 텍스트 노드이거나 요소 노드
- Element.prototype.firstElementChild
  - 첫 번째 자식 요소 노드 반환
  - 요소 노드만 반환
- Element.prototype.lastElementChild
  - 마지막 자식 요소 반환
  - 요소 노드만 반환

3. 자식 노드 존재 확인

- Node.prototype.hasChildNodes( ) 메서드
  - 텍스트 노드를 포함하여 자식 노드의 존재를 확인
  - 불리언 값 반환
- children.length
- Element.prototype.childElementCount

4. 요소 노드의 텍스트 노드 탐색

- firstChild 프로퍼티
  - 텍스트 노드까지 반환하기에

5. 부모 노드 탐색

- Node.prototype.parentNode 프로퍼티
  - 부모노드가 텍스트 노드의 경우는 없다. -> 리프 노드이기에

6. 형제 노드 탐색

- Node.prototype.previousSibling 프로퍼티
  - 부모 노드가 같은 형제 노드 중에서 자신의 이전 형제 노드를 탐색하여 반환
  - 요소 노드 뿐만 아니라 텍스트 노드일 수도 있다.
- Node.prototype.nextSibling 프로퍼티
  - 부모 노드가 같은 형제 노드 중에서 자신의 다음 형제 노드 탐색하여 반환
  - 요소 노드 뿐만 아니라 텍스트 노드일 수도 있다.
- Element.prototype.previousElementSibling 프로퍼티
  - 부모 노드가 같은 형제 노드 중에서 자신의 이전 형제 노드를 탐색하여 반환
  - 요소 노드만 반환
- Element.prototype.nextElementSibling 프로퍼티
  - 부모 노드가 같은 형제 노드 중에서 자신의 다음 형제 노드 탐색하여 반환
  - 요소 노드만 반환

39.4 노드 정보 취득

- Node.prototype.nodeType 프로퍼티
  - 노드 객체의 종류, 즉 노드 타입을 나타내는 상수를 반환
  - 요소 노드 : 1
  - 텍스트 노드 : 3
  - 문서 노드 타입: 9
- Node.prototype.nodeName 프로퍼티
  - 노드의 이름을 문자열로 반환
  - 요소 노드: 대문자 문자열로 태그 이름(UL, LI)을 반환
  - 텍스트 노드: 문자열 "#text" 반환
  - 문서 노드: 문자열 "#document" 반환