JS Deep

DOM

- Browser APIs
  - 웹 브라우저에 내장된 API로 웹 브라우저가 현재 컴퓨터 환경에 관한 데이터를 제공하거나,

    오디오를 재생하는 등 여러가지 유용하고 복잡한 일을 수행할 수 있게 한다.
  - JavaScript로 Browser API들을 사용해서 여러가지 기능을 사용할 수 있다.
- 웹 페이지를 브라우저로 불러오면, 브라우저는 HTML,CSS,JavaScript를 실행 환경인 브라우저 탭에서 실행을 한다.
- JavaScript는 DOM API를 통해 HTML과 CSS를 동적으로 수정, 사용자 인터페이스를 업데이트하는 일에 가장 많이 쓰임
- 문서 객체 모델 (Document Object Model)
- 문서의 구조화된 표현을 제공, 프로그래밍 언어가 DOM구조에 접근할 수 있는 방법을 제공
  - 문서 구조, 스타일, 내용 등을 쉽게 변경할 수 있게 도움
  - HTML 콘텐츠를 추가, 제거, 변경하고, 동적으로 페이지에 스타일을 추가하는 등의 HTML/CSS를 조작할 수 있다.
- HTML 문서를 구조화 하여 각 요소를 객체로 취급
- DOM은 문서를 논리 트리로 표현
- DOM 메서드를 사용하면
  - 프로그래밍적으로 트리에 접근할 수 있고 이를 통해 문서의 구조, 스타일, 컨텐츠를 변경할 수 있다.
- 웹 페이지는 일종의 문서(Document)
- DOM은 동일한 문서를 표현하고, 저장하고, 조작하는 방법을 제공
- DOM은 웹 페이지의 객체 지향 표현이며, JavaScript와 같은 스크립트 언어를 이용해 DOM을 수정할 수 있다

- DOM의 주요 객체
  - window
    - DOM을 표현하는 창
    - 가장 최상위 객체 (작성 시 생략 가능)
    - 탭 기능이 있는 브라우저에서는 각각의 탭을 각각의 window 객체로 나타냄
    - 새 탭 열기: window.open()
    - 경고 대화 상자 표시: window.alert()
    - 인쇄 대화 상자 표시: window.print()
  - document
    - 브라우저가 불러온 웹 페이지

        - 페이지 컨텐츠의 진입점 역할을 하며, < body > 등과 같은 수많은 다른 요소들을 포함하고 있다.
        - 현재 문서의 제목: document.title
        - 현재 문서의 제목 수정: document.title = 'hahaha'
        - document는 window의 속성이다.
    - navigator, location, history, screen 등이 있다.

DOM 조작

- DOM 조작 순서
    1. 선택 (Select)
        - document.querySelector()
            - 제공한 선택자와 일치하는 element 한 개 선택 (첫번째 요소)
            - 제공한 CSS selector를 만족하는 첫 번째 element 객체를 반환(없다면 null)
        - document.querySelectorAll()
            - 제공한 선택자와 일치하는 여러 element를 선택
            - 매칭 할 하나 이상의 셀렉터를 포함하는 유효한 CSS selector를 인자로 받음
            - 제공한 CSS selector를 만족하는 NodeList를 반환
            - NodeList
                - index로만 각 항목에 접근 가능
                - 배열의 forEach메서드 및 다양한 배열 메서드 사용 가능
                - querySelectorAll()에 의해 반환되는 NodeList는 DOM의 변경사항을 실시간으로 반영하지 않음

    2. 조작 (Manipulation)
        - 생성
            - document.createElement()
        - 입력
            - Node.innerText
                - Node 객체와 그 자손의 텍스트 컨텐츠(DOMString)를 표현
                - 사람이 읽을 수 있는 요소만 남김
                - 즉, 줄 바꿈을 인식하고 숨겨진 내용을 무시하는 등 최종적으로 스타일링이 적용된 모습으로 표현됨
        - 추가
            - Node.appendChild()
                - 한 Node를 특정 부모 Node의 자식 NodeList 중 마지막 자식으로 삽입
                - 한번에 오직 하나의 Node만 추가할 수 있음.
                - 추가된 Node 객체를 반환
                - 만약 주어진 Node가 이미 문서에 존재하는 다른 Node를 참조한다면 현재 위치에서 새로운 위치로 이동

        - 삭제
            - Node.removeChild()
                - DOM에서 자식 Node를 제거
                - 제거된 Node를 반환
        - 속성 조회
            - Element.getAttribute()
                - 해당 요소의 지정된 값(문자열)을 반환
                - 인자(attributeName)는 얻고자 하는 속성의 이름
        - 설정
            - Element.setAttribute(name,value)
                - 지정된 요소의 값을 설정
                - 속성이 이미 존재하면 값을 갱신, 존재하지 않으면 지정된 이름과 값으로 새 속성을 추가

Event

- Event object
    - 네트워크 활동이나 사용자와의 상호작용 같은 사건의 발생을 알리기 위한 객체
    - Event 발생
        - 마수으를 클릭하거나 키보드를 누르는 등 사용자 행동으로 발생할 수도 있고
        - 특정 메서드를 호출하여 프로그래밍적으로도 만들어 낼 수 있음
    - Event object
        - DOM 요소는 Event를 받고('수신')
        - 받은 Event를 '처리'할 수 있음
            - Event 처리는 주로 addEventListener() 라는 Event 처리기(Event Handler)를 다양한 html 요소에 '부착'해서 처리

- Event Handler
    - addEventListener()
        - EventTarget.addEventListener(type, listener[, options])
        - 대상에 특정 Event가 발생하면, 할 일을 등록하자
        - EventTarget.addEventListener(type, listener)
        - 지정한 Event가 대상에 전달될 때마다 호출할 함수를 설정
        - Event를 지원하는 모든 객체(Element, Document, Window 등) 를 대상(EventTarget)으로 지정 가능
            - type
                - 반응 할 Event 유형을 나타내는 대소문자 구분 문자열
                - 대표 이벤트
                    - input, click, submit ...

- listener
  - 지정된 타입의 Event를 수신할 객체
  - JavaScript function 객체(콜백 함수)여야 함
  - 콜백 함수는 발생한 Event의 데이터를 가진 Event 객체를 유일한 매개변수로 받음

Event 취소

- event.preventDefault()
  - 현재 Event의 기본 동작을 중단
  - HTML 요소의 기본 동작을 작동하지 않게 막음

---

this

- 전역 문맥에서의 this
  - 브라우저의 전역 객체인 window를 가리킴
  - 전역객체는 모든 객체의 유일한 최상위 객체를 의미
  - console.log(this) // window
- 함수 문맥에서의 this
  - 함수의 this 키워드는 다른 언어와 조금 다르게 동작
  - this의 값은 함수를 호출한 방법에 의해 결정됨
  - 함수 내부에서 this의 값은 함수를 호출한 방법에 의해 좌우됨
  - Nested (Function 키워드) `[주의]`

```
const myobj = {
      number: [1],
      myFunc() {
          console.log(this) // myobj
          this.number.forEach(function(num) {
              console.log(num) // 1
              console.log(this) // window
          } )
          this.number.forEach((num1) => {
              console.log(this) // myobj
          })
          console.log(this.data) // 1
        }
```

```
    }
    myobj.myFunc()
```

- 화살표 함수
  - 화살표 함수는 호출의 위치와 상관없이 상위 스코프를 가리킴(Lexical scope this)
  - 따라서 함수 내의 함수 상황에서는 화살표 함수를 쓰는 것을 권장
  - 하지만 addEventListener에서의 콜백 함수는 특벼라게 function 키워드 사용을 권장