**1. HTML의 form / action 개념**

**1.1 method**

- 특정 양식의 데이터를 action 속성으로 지정한 페이지로 보내는 방식을 정의한 것

**1.2 GET / POST**

**1) GET**

- URL을 통해 이름/값 쌍의 데이터를 전송

- 민감한 데이터에는 사용 x

- 사용자가 결과를 북마크하고자 할 때 유용한 제출 양식

- URL은 최대 3000자 제한

- method 속성의 default 값

**2) POST**

**-** HTTP 요청을 할 때 본문에 직접 데이터를 추가

- 길이 제한 x

**1.3 parameter (name, value)**

**1) GET 코드 예시** (w3school 공식문서 참조)

**텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명**

-> 위와 같이 form 태그에 action 속성은 데이터를 보낼 페이지로, method 속성은 **get**으로 설정한 뒤, input 태그의 name 속성 값과 각 input 태그가 값을 입력받게 설정한다.

**텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명**

-> 다음과 같이 input 태그에 특정 값을 입력하여 제출하면 **URL?name=value** 형태로 데이터가 들어간 것을 볼 수 있다.

**1) POST 코드 예시**

**텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명**

-> get 방식과 마찬가지로 설정한 뒤 method 속성만 **post**로 설정해 준다.

텍스트, 스크린샷, 번호, 디스플레이이(가) 표시된 사진

자동 생성된 설명

-> post 방식으로 보낸 데이터는 URL로는 확인할 수 없지만 네트워크 -> 페이로드에서 확인할 수 있었다.

**2. Javascript**

**2.1 DOM Manipulation**

**1) DOM (Document Object Model)**

HTML, XML 문서를 프로그래밍 언어로 접근할 수 있게 객체 형태로 표현한 모델이다.

Javascript의 DOM API를 사용하여 DOM을 조작할 수 있다.

**2) DOM 조작**

- document**.createElement**(element): 요소 생성

- parent**.append**(child): Node / DOMString 객체 요소를 부모 요소에 자식 요소로서 추가

- .appendChild: append와 동일하나 Node 객체만 추가 가능

- parent**.removeChild**(child): 부모 요소로부터 자식 요소 제거

- parent**.replaceChild**(new,old): 부모 요소의 기존 요소를 새 요소로 변경

- element**.style.**propery = value : 요소의 스타일 변경

- element**.setAttribute**(attribute, value) : 요소의 속성 값 설정

- element**.removeAttribute**(attribute): 요소의 속성 제거

- element**.addEventListener**(event, function): 요소의 특정 행동에 대한 이벤트 추가

**2.2 EventListener**

- DOM 객체에서 특정 이벤트가 발생할 경우 이벤트 처리 핸들러를 추가해주는 객체이다.

- element.addEventListener(event, function): 지정된 요소에 이벤트 핸들러를 연결,

- element.removeEventListener(event, function): 지정된 요소에 연결한 이벤트 핸들러를 제거

-> 동일한 핸들러만 제거 되기에 화살표 함수 등으로 연결된 이벤트 핸들러는 제거 불가능

**2.3 Event Loop**

**1) 이벤트 루프:** 태스크(작업)가 들어오길 기다리고 들어오면 처리 후 또 다음 task가 들어올 때까지 대기하는 자바스크립트의 루프. 단일 호출 스택을 사용하는 자바스크립트 엔진과 상호 연동되어 있다.

**2) 태스크의 종류**

- 외부스크립트가 로드 및 실행

- 이벤트 및 이벤트 핸들러

- setTimeout가 설정된 시간이 되어 콜백 함수 실행

**3) 태스크 큐**

- 비동기 API들이 작업 완료 시 콜백 함수를 대기시키는 큐(FIFO) 형태의 배열

- 이벤트 루프는 호출 스택이 비어질 때마다 혹은 현재 실행 중인 태스크가 없을 때 큐에

대기하고 있는 첫 번째 콜백 함수를 꺼내와 실행시킨다.

**2.4 요소 선택**

**1) getElementById:** 요소의 id 속성을 통해 해당 요소를 선택한다.

 Ex)

**텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명**

**텍스트, 폰트, 화이트, 디자인이(가) 표시된 사진

자동 생성된 설명**

**2) getElementByClassName:** 요소의 class 속성으로 해당 요소를 선택한다. class는 여러곳에서

지정될 수 있으므로 단일 요소가 아닌 HTMLCollection 객체(요소 배열)로 반환된다.

 Ex)

텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

**텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명**

-> myClass를 갖고 있는 요소들이 HTMLCollection 객체로 반환된다.

**3) queryselector vs queryselectorAll**

**- queryselector:** 요소의 선택자를 통해 해당 요소를 선택한다. 첫번째로 일치하는

요소만 반환한다.

**- queryselectorAll:** 선택자와 일치하는 모든 요소를 찾아 NodeList 객체로 반환한다.

텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명 Ex)

**텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명**

**2.5 Node의 속성**

- ELEMENT\_NODE: <div>와 <p>같은 요소

- ATTRIBUTE\_NODE: 요소의 속성 id, class등

- TEXT\_NODE: 요소나 속성 안에 있는 실제 텍스트

- CDATA\_SECTION\_NODE: <!CDATA[[ … ]]>와 같은 xml 메시지의 CData 섹션

- PROCESSING\_INSTRUCTION\_NODE: xml 문서의 processing instruction

- COMMENT\_NODE: 주석 요소

- DOCUMENT\_NODE: 문서

- DOCUMENT\_TYPE\_NODE: <!DOCTYPE html>과 같은 문서의 타입

- DOCUMENT\_FRAGMENT\_NODE: 문서 프래그먼트

**3. Javascript 문법**

**3.1 var, let의 차이**

**1) 선언, 재선언, 할당, 재할당**

**var:** 최초 선언 및 할당 이후에 재선언 및 재할당 모두 가능

선언 전에도 할당 가능 (호이스팅 현상으로 함수 최상위에서 선언된 것으로 간주)

**let: 최초 선언 및 할당 이후** 재할당은 가능하지만 재선언은 불가능

텍스트, 폰트, 스크린샷, 친필이(가) 표시된 사진

자동 생성된 설명 Ex)

**텍스트, 폰트, 스크린샷, 화이트이(가) 표시된 사진

자동 생성된 설명**

**2) 스코프**

**var:** 함수 스코프 혹은 전역 스코프여서 블록 밖에서도 변수에 접근이 가능

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명 **let:** 블록 스코프이므로 블록 밖에서는 접근이 불가능

Ex)

**3.2 const**

- 변화하지 않는 변수, 재선언은 물론 재할당도 불가능

- 기억하기 어려운 값에 대문자 및 밑줄로 구성한 별칭으로도 사용

Ex) 텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

**3.3 ES5의 Array 관련 메소드 (내가 주로 쓰는 것 위주)**

**1) forEach:** 배열의 모든 요소에 대하여 한 번씩 접근, index도 접근 가능

**텍스트, 폰트, 스크린샷, 친필이(가) 표시된 사진

자동 생성된 설명** Ex)

**2) map**

- forEach와 마찬가지로 배열의 모든 요소에 대한 접근 및 수정이 가능

- 단 기존 배열을 수정하는 것이 아닌 새로운 배열을 생성하여 수정한 뒤 반환함

**텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명** Ex)

**3) filter**

텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명- 각 요소에 대하여 접근하여 특정 조건에 달하는 요소만 새로운 배열에 추가하여 반환

Ex)

**4) reduce**

- 각 요소에 대한 누적 계산을 진행하여 결과 값을 반환함

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명 Ex)

**5) IndexOf**

텍스트, 폰트, 라인, 화이트이(가) 표시된 사진

자동 생성된 설명 - 배열의 특정 값의 index를 반환 (index는 0부터 시작)

Ex)

**3.4 rest parameter**

- 이름 앞에 ‘…’을 붙여 정의한 매개변수, 이름 그대로 나머지 인수를 배열로 할당 받음

- 함수의 매개변수로 사용시 반드시 마지막에 사용해야 한다.

Ex) 텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**3.5 Arrow function**

**- 화살표 함수의 형태 및 예시 코드**

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

**+ 화살표 함수의 추가적인 특징**

- 화살표 함수는 this가 없음, 선언 시 외부에서 가져옴

- new 호출도 불가능

- arguments 객체를 지원하지 않음

**3.6 1급 객체**

**- 1급 객체의 기준 및 예시 코드**

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**3.7 Object 개념**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명