window.location

window.location 객체는 현재 문서의 위치(URL)를 나타내며, 브라우저의 위치와 URL을 변경하는 데 사용되는 속성과 메서드를 제공합니다. 이 객체는 웹 페이지의 현재 URL을 가져오거나 설정할 수 있는 다양한 프로퍼티와 메서드를 포함하고 있습니다.

window.location 객체의 주요 프로퍼티

- href: 현재 전체 URL을 문자열 형태로 반환하거나 설정합니다.
- protocol: URL의 프로토콜을 반환하거나 설정합니다(e.g., "http:", "https:").
- host: 호스트 이름과 포트 번호를 반환합니다(e.g., "www.example.com:80").
- hostname: 호스트 이름을 반환합니다(e.g., "www.example.com").
- port: 포트 번호를 반환하거나 설정합니다(e.g., "80", "443").
- pathname: URL의 경로를 반환하거나 설정합니다(e.g., "/path/page").
- search: 쿼리 문자열을 반환하거나 설정합니다(e.g., "?id=123").
- hash: URL의 해시(#) 부분을 반환하거나 설정합니다(e.g., "#section1").

window.location 객체의 주요 메서드

- assign(): 주어진 URL로 이동합니다.
- reload(): 현재 문서를 다시 로드합니다. 매개변수 true를 전달하면 캐시를 무시하고 페이지를 다시 로드합니다.
- replace(): 주어진 URL로 이동하며, 현재 페이지를 브라우저 히스토리에서 제거합니다.

form 태그에 대한 설명

form 태그의 이벤트를 addEventListener로 잡는 방법

태그의 이벤트를 잡기 위해서는 JavaScript의 addEventListener("submit") 메서드를 사용할 수 있습니다. 예를 들어, form 태그의 submit 이벤트를 잡아 처리하고 싶다면 다음과 같이 작성할 수 있습니다:

```
document.querySelector('form').addEventListener('submit', function(event) { });
```

form 태그의 submit을 강제로 막는 방법

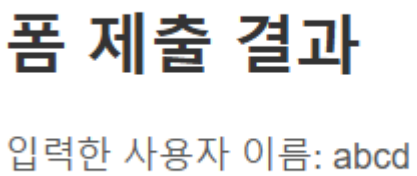
form 태그의 submit을 강제로 막기 위해서는 submit 이벤트에서 event.preventDefault() 메서드를 호출하면 됩니다. 이 메서드는 form이 실제로 서버에 전송되는 것을 방지합니다.

```
event.preventDefault(); // form의 submit을 막음
```

자바스크립트 내에서 자체적으로 submit 하는 방법

자바스크립트 내에서 form을 강제로 제출(submit)하려면 submit() 메서드를 사용할 수 있습니다. 예를 들어, 버튼 클릭 시 form을 제출하려면 다음과 같이 작성할 수 있습니다:

```
document.querySelector('form').submit(); // form을 강제로 제출
```



URLSearchParams란?

URLSearchParams는 URL의 쿼리 스트링을 다루기 위한 JavaScript 인터페이스입니다. 이 인터페이스를 사용하면 URL의 쿼리 파라미터를 손쉽게 추가, 삭제, 수정, 검색할 수 있습니다. 주로 URL의 파라미터를 조작할 때 유용하게 사용됩니다.

기본 사용법

URLSearchParams 객체를 생성하려면 URL 문자열에서 쿼리 스트링을 추출하거나, 직접 쿼리 스트링을 전달하여 생성할 수 있습니다.

```
// URL 문자열에서 쿼리 스트링을 추출하여 URLSearchParams 객체 생성
const url = '[URL]';
const params = new URLSearchParams(url.split('?')[1]);
// 직접 쿼리 스트링을 전달하여 URLSearchParams 객체 생성
const params2 = new URLSearchParams('name=John&age=30');
```

파라미터 추가 및 수정

URLSearchParams 객체에 파라미터를 추가하거나 수정하려면 append() 또는 set() 메서드를 사용합니다.

```
const params = new URLSearchParams();
// 파라미터 추가
params.append('name', 'John');
params.append('age', '30');
// 파라미터 수정
params.set('name', 'Jane');
```

파라미터 검색

URLSearchParams 객체에서 특정 파라미터의 값을 검색하려면 get() 메서드를 사용합니다. 또한, 특정 파라미터가 존재하는지 확인하려면 has() 메서드를 사용합니다.

```
const url = '[URL]';

const params = new URLSearchParams(url.split('?')[1]);

// 파라미터 값 검색

const name = params.get('name'); // 'John'

const age = params.get('age'); // '30'

// 파라미터 존재 여부 확인

const hasName = params.has('name'); // true

const hasGender = params.has('gender'); // false
```

파라미터 삭제

URLSearchParams 객체에서 특정 파라미터를 삭제하려면 delete() 메서드를 사용합니다.

```
const params = new URLSearchParams('name=John&age=30');

// 파라미터 삭제

params.delete('age');

console.log(params.toString()); // 'name=John'
```

모든 파라미터 조회

URLSearchParams 객체의 모든 파라미터와 그 값을 조회하려면 forEach() 메서드를 사용합니다.

```
const params = new URLSearchParams('name=John&age=30');

params.forEach((value, key) => {

console.log(`${key}: ${value}`);

});

// 출력:

// name: John

// age: 30
```

문자열로 변환

URLSearchParams 객체를 문자열로 변환하려면 toString() 메서드를 사용합니다. 이 메서드는 URL 인코딩된 쿼리 스트링을 반환합니다.

```
const params = new URLSearchParams();

params.append('name', 'John');

params.append('age', '30');

const queryString = params.toString();

console.log(queryString); // 'name=John&age=30'
```