

# Session 16

Local Storage & 비동기 Todo List

NEXT X LIKELION 문지후

# 목차

## 0. JavaScript 보충

JavaScript는 참 ...

구조분해할당이란?

비동기 복습

Modal 만들기(DOM요소 다루기 + 이벤트 처리 복습2)

## 1. Local Storage 개념

쿠키 & 웹 스토리지

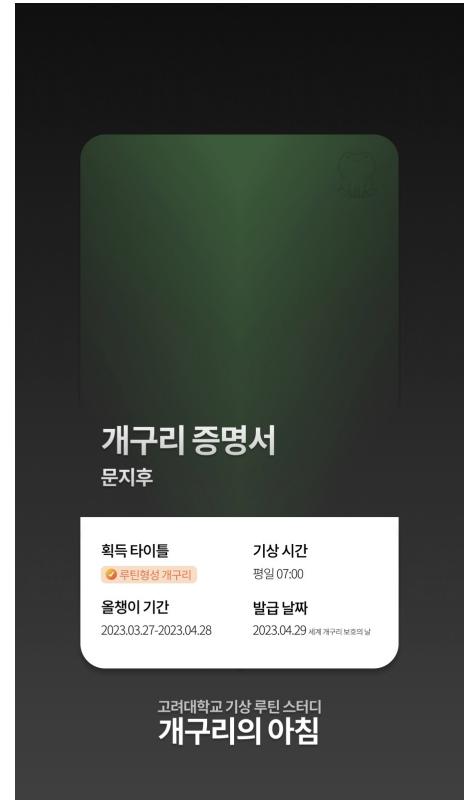
기본 사용법

## 2. Todo List 실습

# 스몰톡



- [개구리의 아침2 ...](#)
- [컴포트존 인사이트 ...](#)



# 0. JavaScript 보충

---

JavaScript는 참 ...

구조분해할당이란?

비동기 복습

Modal 만들기(DOM요소 다루기 + 이벤트 처리 복습2)

# JavaScript는 참...

JavaScript 보충

## Primitive Type 기본형

Number / String / Boolean / Null / Undefined ...

X  
10

y  
20

값을 복사해서  
담는 형식

The screenshot shows a browser's developer tools JS console. It has tabs for HTML, CSS, and JS, with JS selected. The code area contains the following script:

```
1 let x = 10;
2 let y = x;
3
4 console.log(x);
5 console.log(y);
6
7 y = 20;
8
9 console.log(x);
10 console.log(y);
```

The Console output shows the following results:

```
10
10
10
20
```

## Reference Type 참조형

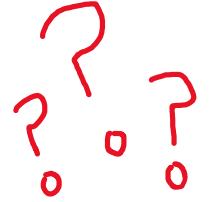
Object (배열도 객체 ...!)

#0x00 (주소)

‘지후’      ‘24’  
‘happy’  
‘교육학과 전공’

jihoo1      jihoo2  
#0x00      #0x00

참조할 수 있는  
주소값을 담는 형식



The screenshot shows a browser's developer tools JS console. It has tabs for JS, CSS, and JS, with JS selected. The code area contains the following script:

```
1 let jihoo1 = ['지후', '24', 'happy'];
2 let jihoo2 = jihoo1;
3
4 console.log(jihoo1);
5 console.log(jihoo2);
6
7 jihoo2.push('교육학과 전공');
8
9 console.log(jihoo1);
10 console.log(jihoo2);
```

The Console output shows the following results:

```
// [object Array] (3)
["지후", "24", "happy"]

// [object Array] (3)
["지후", "24", "happy"]

// [object Array] (4)
["지후", "24", "happy", "교육학과 전공"]

// [object Array] (4)
["지후", "24", "happy", "교육학과 전공"]
```

The screenshot shows a browser's developer tools JS console. It has tabs for JS, CSS, and JS, with JS selected. The code area contains the following script:

```
1 let jihoo1 = ['지후', '24', 'happy'];
2 let jihoo2 = jihoo1;
3
4 console.log(jihoo1);
5 console.log(jihoo2);
6
7 jihoo2.push('교육학과 전공');
8
9 console.log(jihoo1);
10 console.log(jihoo2);
```

The Console output shows the following results:

```
// [object Array] (3)
["지후", "24", "happy"]

// [object Array] (3)
["지후", "24", "happy"]

// [object Array] (4)
["지후", "24", "happy", "교육학과 전공"]

// [object Array] (4)
["지후", "24", "happy", "교육학과 전공"]
```

# JavaScript는 참...

JavaScript 보충

그럼 배열값 복사하고 변경하려면 어떡해요?

[1] slice메소드를 사용해보자

```
① JS
1 let jihoo1 = ['지후', '24', 'happy'];
2 let jihoo2 = jihoo1.slice();
3
4 console.log(jihoo1);
5 console.log(jihoo2);
6
7 jihoo2.push('교육학과 전공');
8
9 console.log(jihoo1);
10 console.log(jihoo2);
11
```

Console

```
// [object Array] (3)
["지후", "24", "happy"] jihoo1
// [object Array] (3)
["지후", "24", "happy"] jihoo2
// [object Array] (3)
["지후", "24", "happy"] jihoo1
// [object Array] (4)
["지후", "24", "happy", "교육학과 전공"] jihoo2
```



## slice메소드에 대해

slice() 메서드는 어떤 배열의 begin부터 end 까지(end 미포함)에 대한 얇은 복사본을 새로운 배열 객체로 반환합니다. 원본 배열은 바뀌지 않습니다.

[https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Global\\_Objects/Array/slice](https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Global_Objects/Array/slice)

# JavaScript는 참...

JavaScript 보충

그럼 배열값 복사하고 변경하려면 어떡해요?

[2] **spread(전개 구문)**를 사용해보자



```
1 let jihoo1 = ['지후', '24', 'happy'];
2 let jihoo2 = [...jihoo1, '교육학과 전공'];
3
4 console.log(jihoo1);
5 console.log(jihoo2);
6
```



## spread문법에 대해

전개 구문을 사용하면 배열이나 문자열과 같이 반복 가능한 문자를 0개 이상의 인수 (함수로 호출할 경우) 또는 요소 (배열 리터럴의 경우)로 확장하여, 0개 이상의 키-값의 쌍으로 객체로 확장시킬 수 있습니다.

[https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Operators/Spread\\_syntax](https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Operators/Spread_syntax)

Console

```
// [object Array] (3)
["지후", "24", "happy"]
```

jihoo1

```
// [object Array] (4)
["지후", "24", "happy", "교육학과 전공"]
```

jihoo2

→ jihoo1의 요소들을 ‘펼쳐주는’(전개해주는) 역할

# JavaScript는 참...

JavaScript 보충

## 템플릿 리터럴 Template literal

작은 따옴표('') 또는 큰 따옴표("") 대신 백틱(``)을 사용한 문자열 표기법



JS

```
1 const num1 = 10;
2 const num2 = 20;
3 console.log(`num1 + num2 = ${num1+num2}입니다.`);
```

Console  
"num1 + num2 = 30입니다."

표현식을 편리하게 사용

JS

```
1 const propose = `사실 문지후는 ${20+4}살이고요
2
3 요즘 먹고 싶은 것 다 먹어서 큰일이고요
4
5 언젠가 운동하겠죠`;
6
7 console.log(propose);
```

Console  
"사실 문지후는 24살이고요  
요즘 먹고 싶은 것 다 먹어서 큰일이고요  
언젠가 운동하겠죠"

일반 문자열과 달리 줄바꿈(개행) 허용

# 구조 분해 할당이란?

JavaScript 보충

개념

## 구조/분해/할당

구조를 / 분해해서 / 할당하는 것

배열이나 객체의 속성을 해체하여,  
그 값을 개별 변수에 담을 수 있게 하는 JavaScript 표현식 (Mdn문서)

[https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Operators/Destructuring\\_assignment](https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Operators/Destructuring_assignment)

🤔 생각해보기

## 왜 필요한데요?

The screenshot shows a browser's developer tools console interface. On the left, there are tabs for HTML, CSS, and JS. The JS tab is active, showing the following code:

```
const people = ['혜영', '정윤', '범진'];
const pizza = people[0];
const chicken = people[1];
const sandwich = people[2];
console.log(pizza);
console.log(chicken);
console.log(sandwich);
```

Below the code, the console output is displayed:

```
"혜영"
"정윤"
"범진"
```

# 구조 분해 할당 (배열)

JavaScript 보충

🤔 생각해보기

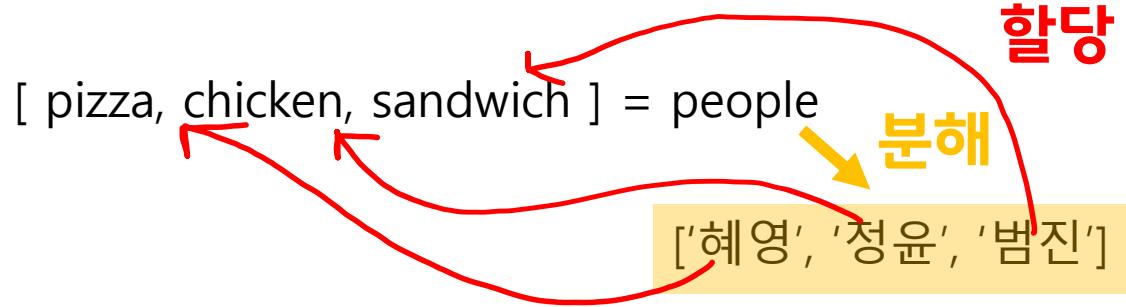
JS

```
1 const people = ['혜영', '정윤', '범진'];
2
3 const pizza = people[0];
4 const chicken = people[1];
5 const sandwich = people[2];
6
7 console.log(pizza);
8 console.log(chicken);
9 console.log(sandwich);
```

Console

```
"혜영"
"정윤"
"범진"
```

구조분해할당 활용 X



Session16 JS 1 unsaved changes X

```
1 const people = ['혜영', '정윤', '범진'];
2
3 [pizza, chicken, sandwich] = people;
4
5 console.log(pizza);
6 console.log(chicken);
7 console.log(sandwich);
```

Console

```
"혜영"
"정윤"
"범진"
```

구조분해할당 활용 O

NEXT X LIKELION

# 구조 분해 할당 (배열)

JavaScript 보충



## 기본 사용법

```
[pizza, chicken, sandwich] = people;
```

할당 받을 변수 = 분해해서 왼쪽에 할당할 값



할당 받는 변수와 할당하는 값의 길이가 일치하지 않으면?

할당하는 값이 더 적을 때

```
JS
1 const people = ['혜영', '정윤'];
2
3 [pizza, chicken, sandwich] = people;
4
5 console.log(pizza);
6 console.log(chicken);
7 console.log(sandwich);
```

Console

```
"혜영"
"정윤"
undefined
```

오류가 나지 않고,  
남아있는 변수가 undefined로 채워짐

할당하는 값이 더 많을 때

```
JS 3 unsaved changes ×
1 const people = ['혜영', '정윤', '범진', '지후', '도라에몽'];
2
3 [pizza, chicken, sandwich] = people;
4
5 console.log(pizza);
6 console.log(chicken);
7 console.log(sandwich);
```

Console

```
"혜영"
"정윤"
"범진"
```

오류가 나지 않고,  
길이에 맞춰서 할당됨

NEXT X LIKELION

# 구조 분해 할당 (배열)

JavaScript 보충

## 심화 사용법

Swap할 때(값을 교환할 때) 편리하게 사용

기존 방식 (임시 변수에 담아서 사용)

JS

```
1 let Monday = '세션';
2 let Tuesday = '해커톤';
3
4 console.log('--변경 전--');
5 console.log(Monday);
6 console.log(Tuesday);
7
8 let temp = Monday;
9 Monday = Tuesday;
10 Tuesday = temp;
11
12 console.log('--변경 후--');
13 console.log(Monday);
14 console.log(Tuesday);
```



구조 분해 할당 활용

JS

```
1 let Monday = '세션';
2 let Tuesday = '해커톤';
3
4 console.log('--변경 전--');
5 console.log(Monday);
6 console.log(Tuesday);
7
8 [Monday, Tuesday] = ['해커톤', '세션'];
9
10 console.log('--변경 후--');
11 console.log(Monday);
12 console.log(Tuesday);
13
```

# 구조 분해 할당 (배열)

JavaScript 보충



## 심화 사용법

남는 변수들을 위한 default값 설정  
(선언 변수 개수 > 할당하는 값 개수)

```
JS
1
2 const people = ['혜영', '정윤'];
3
4 [pizza, chicken, sandwich = '미정'] = people;
5
6 console.log(pizza);
7 console.log(chicken);
8 console.log(sandwich);
9
10
```

Console

```
"혜영"
"정윤"
"미정"
```

남는 값들을 위한 설정  
(선언 변수 개수 < 할당하는 값 개수)

```
JS
1
2 const people = ['혜영', '정윤', '범진', '지후', '도라에몽'];
3
4 [pizza, chicken, ...sandwich] = people;
5
6 console.log(pizza);
7 console.log(chicken);
8 console.log(sandwich);
9
10
```

Console

```
"혜영"
"정윤"
// [object Array] (3)
["범진", "지후", "도라에몽"]
```

# 구조 분해 할당 (객체)

JavaScript 보충



## 기본 사용법

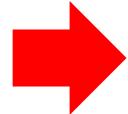
HTML CSS JS

```
1 const jihoo = {  
2   name: '문지후',  
3   age: 24,  
4   state: '달리는 중',  
5   home: '신당'  
6 };  
7  
8 const name = jihoo.name;  
9 const age = jihoo.age;  
10  
11 console.log(name);  
12 console.log(age);  
13
```

Console

```
"문지후"
```

24



HTML CSS JS

```
1 const jihoo = {  
2   name: '문지후',  
3   age: 24,  
4   state: '달리는 중',  
5   home: '신당'  
6 };  
7  
8 const {name, age} = jihoo;  
9  
10 console.log(name);  
11 console.log(age);  
12
```

Console

```
"문지후"
```

24

할당 받을 변수 = 분해해서 왼쪽에 할당할 값

- 배열과 달리 중괄호{} 활용
- 할당하는 객체에서, 왼쪽에 선언된 변수이름과 똑같은 프로퍼티명이 있으면, 그 값이 할당되는 방식

프로퍼티명과 다른 새로운 변수명으로 할당받고 싶다면?

```
const {name: myName, age} = jihoo;
```

- 프로퍼티명: 새로운 변수이름 형식으로 작성

# Modal 만들기

Modal 만들기(DOM요소 다루기 + 이벤트 처리 복습2)

## Modal(모달)이란?

기존의 브라우저 페이지 위에 새로운 창이 아닌, 레이어를 까는 것  
(새로운 페이지로 이동할 필요 없이 편리하게 정보 표시 가능)

☞ 더 알아보기: <https://brunch.co.kr/@bcc5736f7b26444/1>

Open Vanilla JavaScript Animation Modal

## 요구사항

- 1) 클릭했을 때 모달창이 생기게 만들기
- 2) X 버튼 눌렀을 때 모달창 없어지게 만들기
- 3) 배경 눌렀을 때 모달창 없어지게 만들기

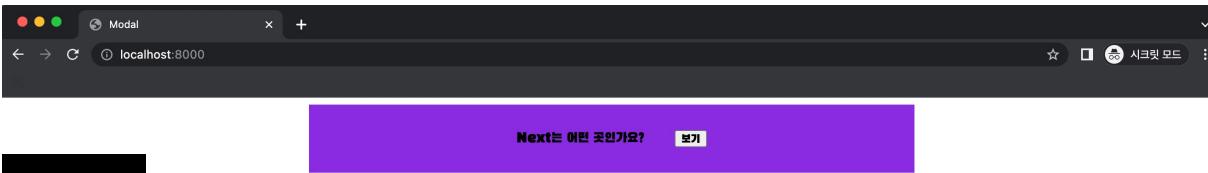
# Modal 만들기

Modal 만들기(DOM요소 다루기 + 이벤트 처리 복습2)

## Modal 활성화 컨트롤 방식

[보기] 버튼을 클릭하기 전에는 모달이 보이지 않아야 한다.

[보기] 버튼 클릭하지 않았을 때



html

```
<div class="modal hidden"></div>
```

모달의 class 이름을 modal과 hidden으로 주고

css

```
.hidden {  
    display: none;  
}
```

평소에 hidden일 때는 보이지 않게 설정한다.

html

```
<div class="modal hidden"></div>
```

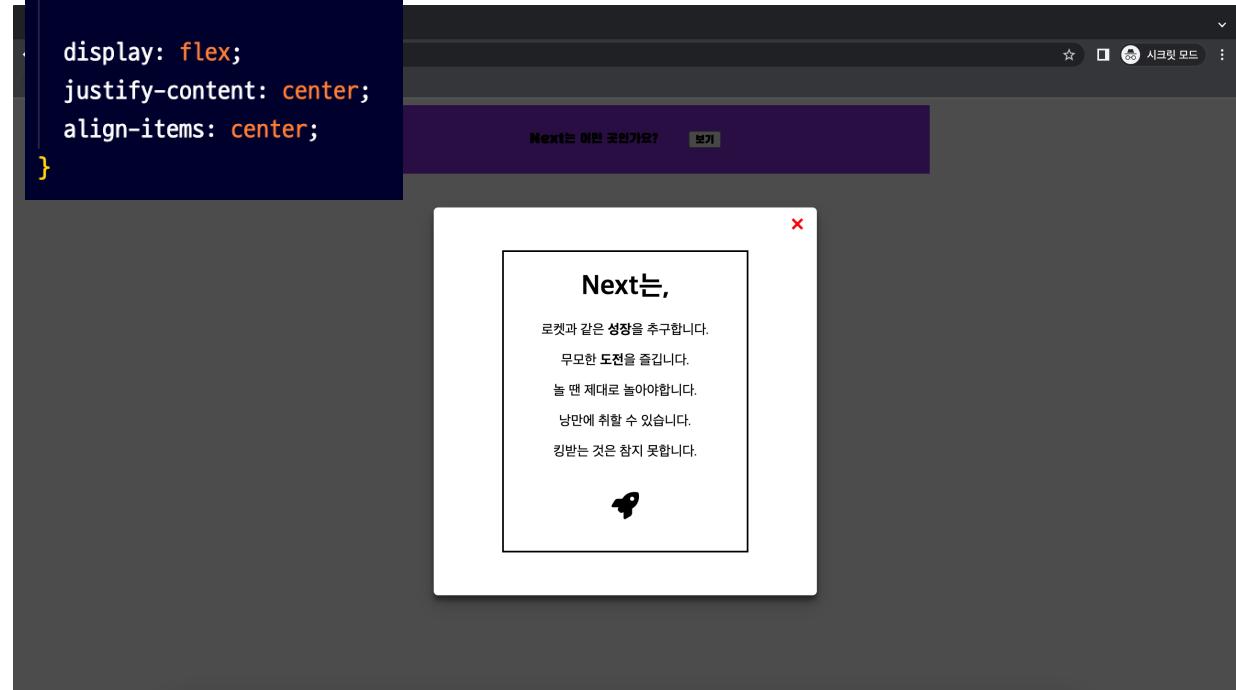
모달의 class 이름에서 hidden을 없애준다.

css

```
.modal {  
    position: fixed;  
    top: 0;  
    left: 0;  
    width: 100%;  
    height: 100%;  
  
    display: flex;  
    justify-content: center;  
    align-items: center;  
}
```

modal일 때는 보이도록 설정한다.

[보기] 버튼 클릭했을 때



NEXT X LIKELION

# Modal 만들기

Modal 만들기(DOM요소 다루기 + 이벤트 처리 복습2)

[보기] 버튼 클릭했을 때 보이도록 만들기 - 활성화시킬 버튼의 DOM요소에 접근

Next는 어떤 곳인가요?

보기

html

```
<div class="container">
  <div>Next는 어떤 곳인가요?</div>
  <button class="open-button">보기</button>
</div>

<div class="modal hidden">
  <div class="modal-content">
    <div class="box">
      <h1>Next는,</h1>
      <p>로켓과 같은 <b>성장</b>을 추구합니다.</p>
      <p>무모한 <b>도전</b>을 즐깁니다.</p>
      <p>놀 땐 제대로 놀아야합니다.</p>
      <p>낭만에 취할 수 있습니다.</p>
      <p>킹받는 것은 참지 못합니다.</p>
      <i class="fas fa-duotone fa-rocket fa-bounce fa-2xl"></i>
      <button class="close">X</button>
    </div>
  </div>
<div class="modal-overlay"></div>
</div>
```

main.js

```
const modal = document.querySelector(".modal");
```

```
/* 모달 여는 버튼 Dom요소에 접근하기*/
```

```
const openBtn = ?
```

# Modal 만들기

Modal 만들기(DOM요소 다루기 + 이벤트 처리 복습2)

[보기] 버튼 클릭했을 때 보이도록 만들기 - 활성화시킬 때 실행할 함수 정의하기

Next는 어떤 곳인가요?

보기

html

```
<div class="container">
  <div>Next는 어떤 곳인가요?</div>
  <button class="open-button">보기</button>
</div>
<div class="modal hidden">
  <div class="modal-content">
    <div class="box">
      <h1>Next는,</h1>
      <p>로켓과 같은 <b>성장</b>을 추구합니다.</p>
      <p>무모한 <b>도전</b>을 즐깁니다.</p>
      <p>놀 땐 제대로 놀아야합니다.</p>
      <p>낭만에 취할 수 있습니다.</p>
      <p>킹받는 것은 참지 못합니다.</p>
      <i class="fas fa-duotone fa-rocket fa-bounce fa-2xl"></i>
      <button class="close">X</button>
    </div>
  </div>
<div class="modal-overlay"></div>
</div>
```

main.js

```
const modal = document.querySelector(".modal");

/* 모달 여는 버튼 Dom요소에 접근하기 */
const openBtn = document.querySelector(".open-button");

/* 모달 열 때의 실행할 함수 정의하기 */

function openModal() {
  ?
}
```

# Modal 만들기

Modal 만들기(DOM요소 다루기 + 이벤트 처리 복습2)

[보기] 버튼 클릭했을 때 보이도록 만들기 - 버튼 클릭했을 때 보이도록 만들기

Next는 어떤 곳인가요?

보기

html

```
<div class="container">
  <div>Next는 어떤 곳인가요?</div>
  <button class="open-button">보기</button>
</div>
<div class="modal hidden">X
  <div class="modal-content">
    <div class="box">
      <h1>Next는,</h1>
      <p>로켓과 같은 <b>성장</b>을 추구합니다.</p>
      <p>무모한 <b>도전</b>을 즐깁니다.</p>
      <p>놀 땐 제대로 놀아야합니다.</p>
      <p>낭만에 취할 수 있습니다.</p>
      <p>킹받는 것은 참지 못합니다.</p>
      <i class="fas fa-duotone fa-rocket fa-bounce fa-2xl"></i>
      <button class="close">X</button>
    </div>
  </div>
<div class="modal-overlay"></div>
</div>
```

main.js

```
const modal = document.querySelector(".modal");

/* 모달 여는 버튼 Dom요소에 접근하기 */
const openBtn = document.querySelector(".open-button");

/* 모달 열 때의 실행할 함수 정의하기 */

function openModal() {
  modal.classList.remove("hidden");
}

/* 버튼 클릭했을 때 모달 활성화하는 함수 실행하기 */
```

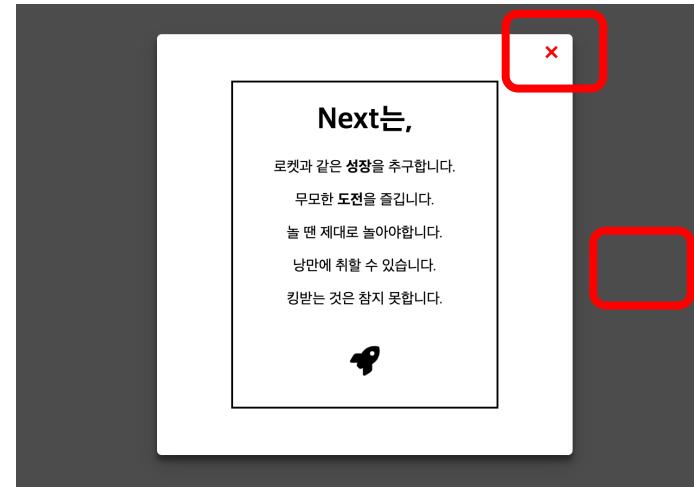
?

LION

# Modal 만들기

Modal 만들기(DOM요소 다루기 + 이벤트 처리 복습2)

X버튼, 검은 바탕 클릭했을 때 모달창 보이지 않게 만들기  
- 비활성화 시킬 버튼, 바탕의 DOM요소에 접근



## html

```
<div class="container">
  <div>Next는 어떤 곳인가요?</div>
  <button class="open-button">보기</button>
</div>
<div class="modal hidden">
  <div class="modal-content">
    <div class="box">
      <h1>Next는,</h1>
      <p>로켓과 같은 <b>성장</b>을 추구합니다.</p>
      <p>무모한 <b>도전</b>을 즐깁니다.</p>
      <p>늘 땐 제대로 놀아야합니다.</p>
      <p>낭만에 취할 수 있습니다.</p>
      <p>킹받는 것은 참지 못합니다.</p>
      <i class="fas fa-duotone fa-rocket fa-bounce fa-2xl"></i>
      <button class="close">X</button>
    </div>
  </div>
<div class="modal-overlay"></div>
</div>
```

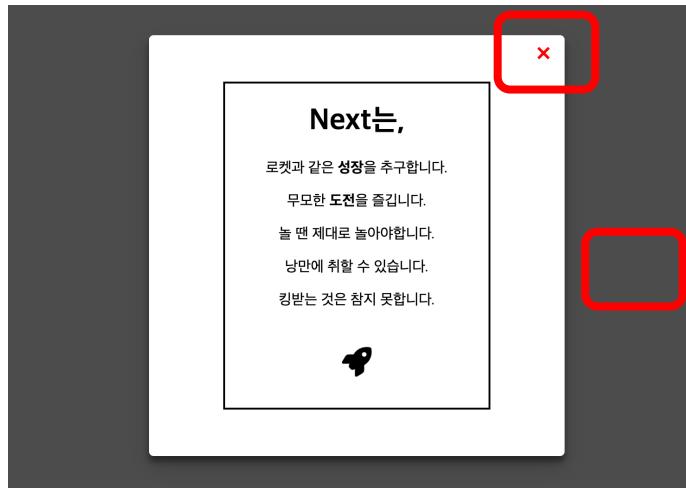
## main.js

```
/* 모달 닫는 Dom요소에 접근하기 */
const closeBtn = modal.querySelector(".close");
const overlay = modal.querySelector(".modal-overlay");
```

# Modal 만들기

Modal 만들기(DOM요소 다루기 + 이벤트 처리 복습2)

X버튼, 검은 바탕 클릭했을 때 모달창 보이지 않게 만들기  
- 비활성화시킬 때 실행할 함수 정의하기



활성화할 때 없었던 class인 hidden을 다시 추가해주어야 한다.

html

```
<div class="container">
  <div>Next는 어떤 곳인가요?</div>
  <button class="open-button">보기</button>
</div>


<div class="modal-content">
    <div class="box">
      <h1>Next는,</h1>
      <p>로켓과 같은 <b>성장</b>을 추구합니다.</p>
      <p>무모한 <b>도전</b>을 즐깁니다.</p>
      <p>늘 땐 제대로 놀아야합니다.</p>
      <p>낭만에 취할 수 있습니다.</p>
      <p>킹받는 것은 참지 못합니다.</p>
      <i class="fas fa-duotone fa-rocket fa-bounce fa-2xl"></i>
      <button class="close">X</button>
    </div>
  </div>
<div class="modal-overlay"></div>
</div>


```

main.js

```
/* 모달 닫는 Dom요소에 접근하기 */
const closeBtn = modal.querySelector(".close");
const overlay = modal.querySelector(".modal-overlay");

/* 모달 닫을 때의 실행할 함수 정의하기 */

function closeModal() {
  ?
}
```

# Modal 만들기

Modal 만들기(DOM요소 다루기 + 이벤트 처리 복습2)

X버튼, 검은 바탕 클릭했을 때 모달창 보이지 않게 만들기  
- 클릭했을 때 보이지 않게 만들기

html

```
<div class="container">
  <div>Next는 어떤 곳인가요?</div>
  <button class="open-button">보기</button>
</div>
<div class="modal hidden">
  <div class="modal-content">
    <div class="box">
      <h1>Next는,</h1>
      <p>로켓과 같은 <b>성장</b>을 추구합니다.</p>
      <p>무모한 <b>도전</b>을 즐깁니다.</p>
      <p>늘 땐 제대로 놀아야합니다.</p>
      <p>낭만에 취할 수 있습니다.</p>
      <p>킹받는 것은 참지 못합니다.</p>
      <i class="fas fa-duotone fa-rocket fa-bounce fa-2xl"></i>
      <button class="close">X</button>
    </div>
  </div>
  <div class="modal-overlay"></div>
</div>
```

main.js

```
/* 모달 닫는 Dom요소에 접근하기 */
const closeBtn = modal.querySelector(".close");
const overlay = modal.querySelector(".modal-overlay");

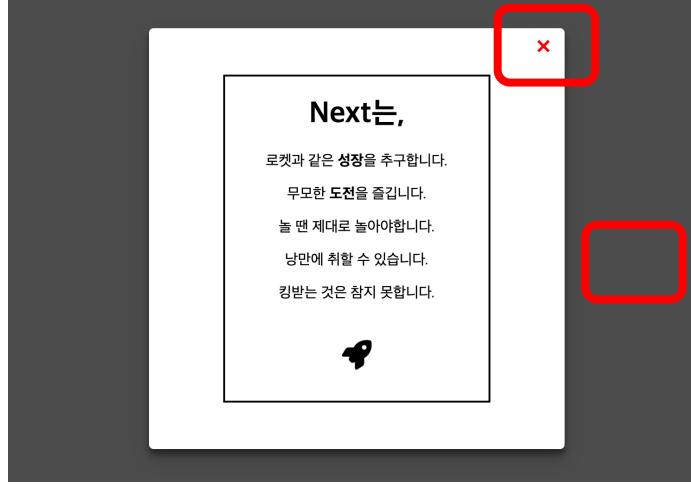
/* 모달 닫을 때의 실행할 함수 정의하기 */

function closeModal() {
  modal.classList.add("hidden");
}

/* 버튼 클릭했을 때 모달 비활성화하는 함수 실행하기 */

closeBtn.addEventListener("click", closeModal);
overlay.addEventListener("click", closeModal);
```

NEXT X LIKELION



# 1. Local Storage 개념

---

쿠키 & 웹 스토리지

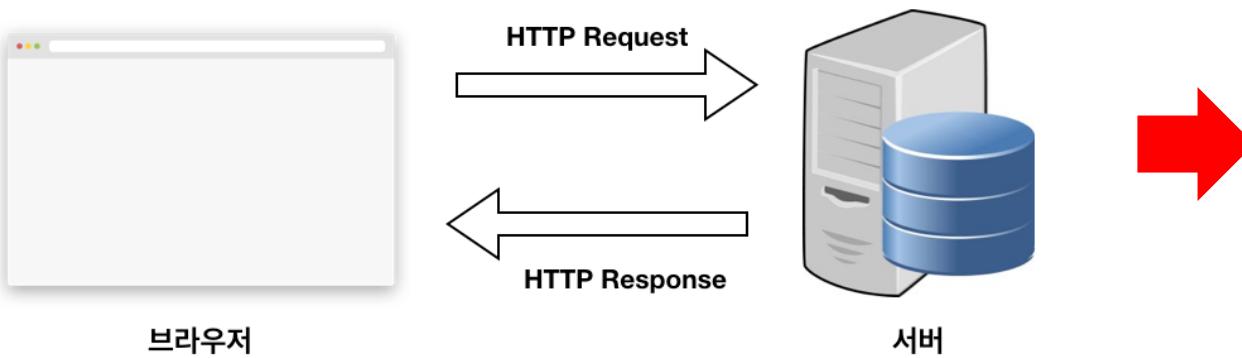
기본 사용법

# 쿠키와 웹 스토리지

Local Storage 개념

## 왜 필요할까?

### [ 기존 방식 ]



불필요하게 서버와 많이 통신하며 **트래픽 부담**이 상승

사소한 부분에서도 로딩이 생기며 **UX 저하**

HTTP 프로토콜은 클라이언트와 서버가 데이터를 주고받을 때,  
서로의 데이터를 기억하지 않고 + 연결은 계속해서 끊긴다

**Connectionless** 클라이언트가 요청한 후 응답을 받으면 연결을 끊는다

**Stateless** 통신이 끝나면 상태를 유지하지 않는다

( = 첫 번째 통신에서 데이터를 주고 받았다고 해도, 두 번째 통신에서 이전 데이터를 유지하지 않는다)

# 쿠키와 웹 스토리지

Local Storage 개념

언제 쓰일까?

NAVER

PC방 등 공용PC라면 QR코드 로그인이 더 안전해요. ×

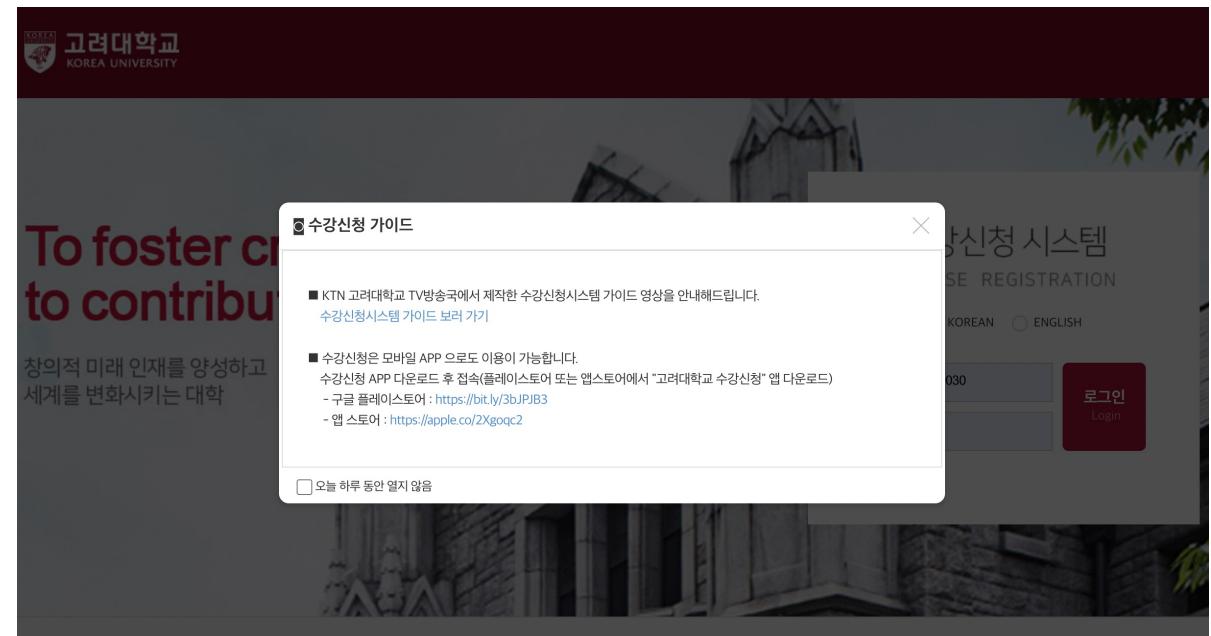
ID 로그인     일회용 번호     QR코드

아이디  
 비밀번호

로그인 상태 유지    IP보안

**로그인**

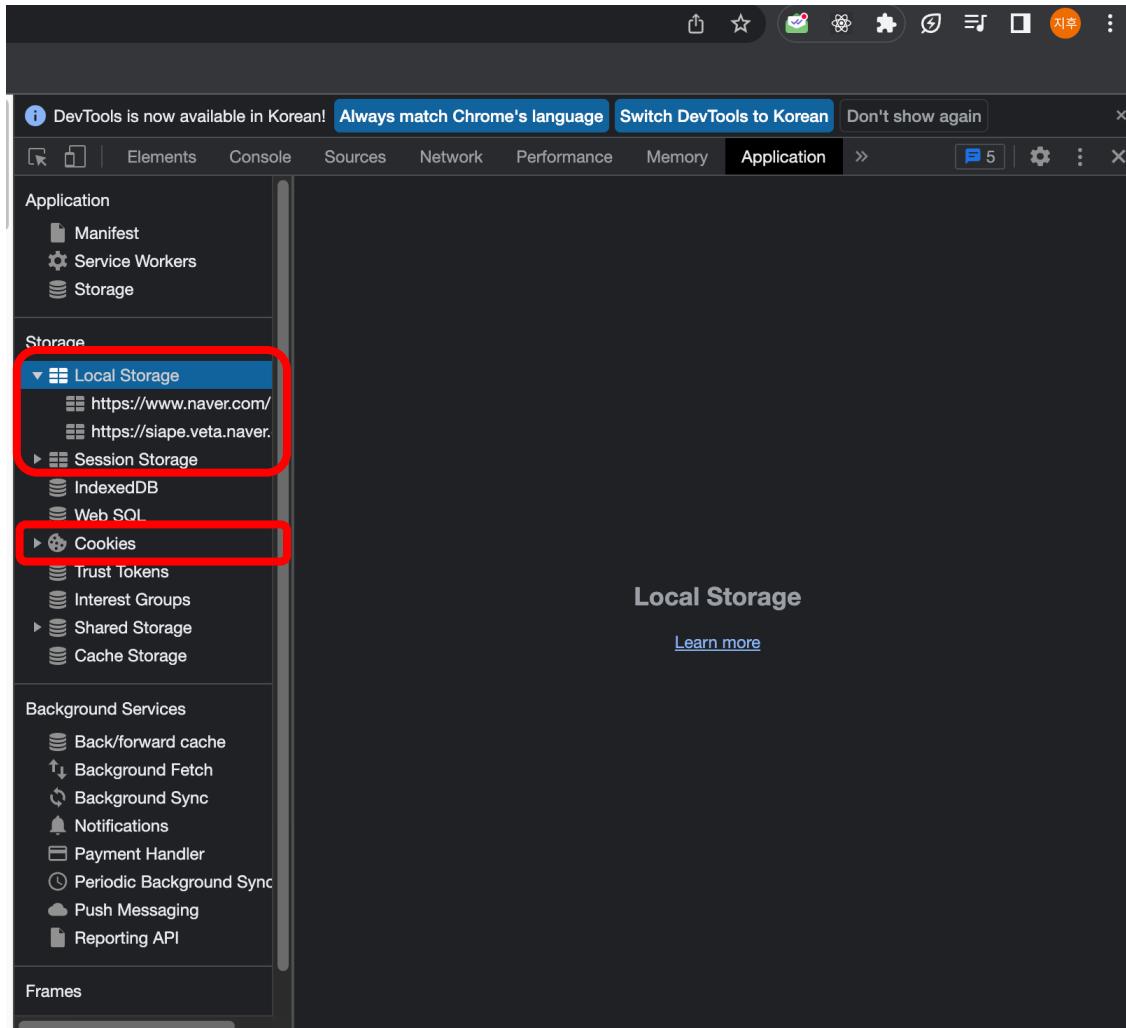
[비밀번호 찾기](#) | [아이디 찾기](#) | [회원가입](#)



# 쿠키와 웹 스토리지

Local Storage 개념

직접 확인해보기



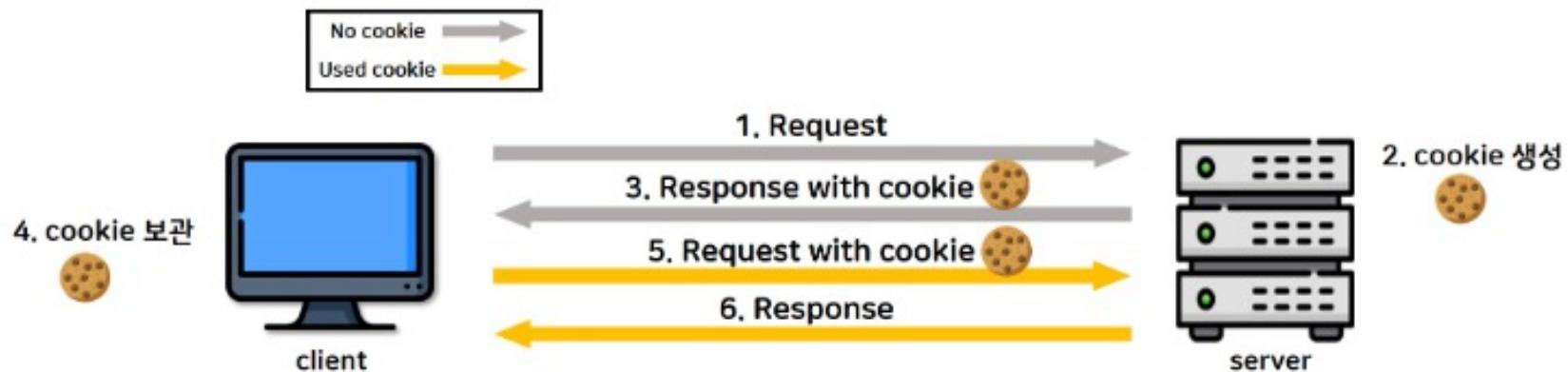
# 쿠키와 웹 스토리지

Local Storage 개념

## 쿠키 VS 웹 스토리지

서버가 클라이언트에게 전송하는 작은 데이터

(이름, 값, 도메인 정보, 경로 정보, 만료 일자, 시간 등이 담겨 있음)



- 매번 서버에 전송
- 저장 용량이 작음
- 보안이 취약

# 쿠키와 웹 스토리지

Local Storage 개념

## 쿠키 VS 웹 스토리지

웹 브라우저가 직접 데이터를 저장할 수 있게 해주는  
(서버가 아닌 클라이언트에 저장할 수 있게 해주는)  
HTML5의 기능

지속성에 따라 구분

### Session Storage 세션스토리지

하나의 세션(session)만을 위한 데이터 저장  
(사용자가 브라우저 탭이나 창을 닫을 때 데이터 사라짐)

### cf) 세션(session)이란?

세션(Session)이란 HTTP 프로토콜을 사용하는 인터넷 사용자가 어떤 웹사이트를 방문할 경우, 웹 사이트의 여러 페이지에 걸쳐 사용되는 사용자 정보를 저장하는 방법을 의미.  
사용자가 브라우저를 닫아 서버와의 연결을 끝내는 시점까지  
를 세션이라고 칭함.

### Local Storage 로컬스토리지

보관 기한이 없는 데이터 저장  
(브라우저 탭 또는 창이 닫히거나, 컴퓨터를 재부팅해도 사라지지 않음)

# 쿠키와 웹 스토리지

Local Storage 개념

## 쿠키 VS 웹 스토리지 총정리

	Cookie	Session Storage (Web Storage)	Local Storage (Web Storage)
개당 사용 가능 용량	최대 4KB	평균 5MB	평균 5MB
유지 기간	설정 가능	한 세션 (탭 닫을 시 내용 제거)	사용자가 삭제하기 전까지
통신 여부	서버와 계속 연결	설정 가능	설정 가능
사용처	기간 설정 필요한 부분 (팝업 창)	보안을 필요로 하는 경우 (은행 로그인 등)	상시 저장이 필요한 경우 (로그인 유지 등)

# 기본 사용법

Local Storage 개념

## Local Storage 기본 사용법

Local Storage에 아이템 추가

```
window.localStorage.setItem(key, value)
```

Local Storage의 아이템 읽기

```
window.localStorage.getItem(key);
```

Local Storage의 아이템 삭제

```
window.localStorage.removeItem(key);
```

# 기본 사용법

## Local Storage 예제

## Local Storage 기본 사용법 - 개발자 도구 console창에 작성해서 바로 확인해보자

Local Storage에 아이템 추가  
key - value 형태로 전달하여 추가

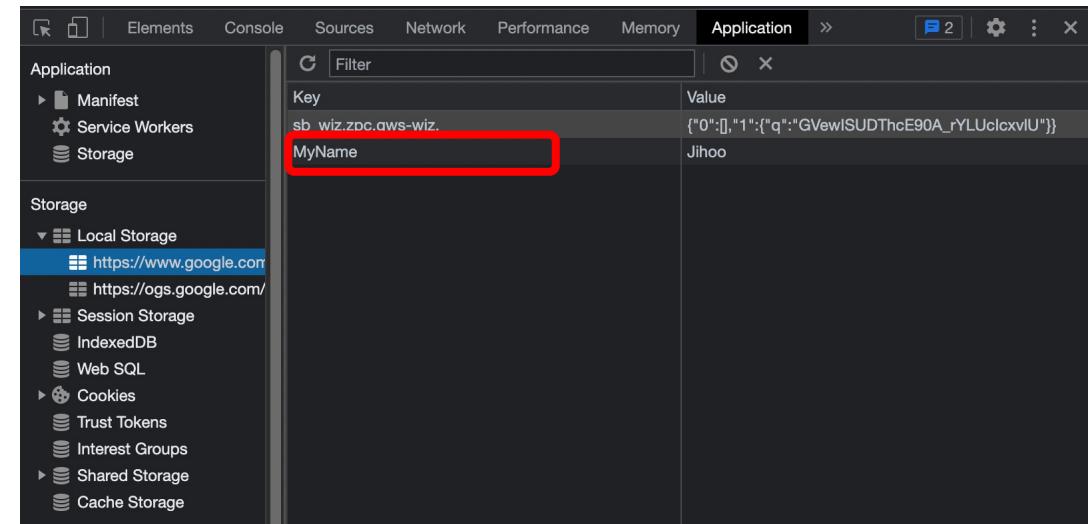
## Local Storage의 아이템 읽기 key로 조회

## Local Storage의 아이템 삭제 Key로 조회하여 삭제

A screenshot of the Chrome DevTools interface, specifically the Console tab. The top navigation bar includes tabs for Elements, Console (which is selected), Sources, Network, Performance, Memory, Application, and more. Below the tabs is a toolbar with icons for back, forward, refresh, and search. The main area shows a command-line interface with the following history:

```
> window.localStorage.setItem('MyName', 'Jihoo')
<- undefined
> window.localStorage.getItem('MyName')
<- 'Jihoo'
> window.localStorage.removeItem('MyName')
<- undefined
> |
```

The status bar at the bottom indicates there are 2 issues.



## 2. Todo List 실습

---

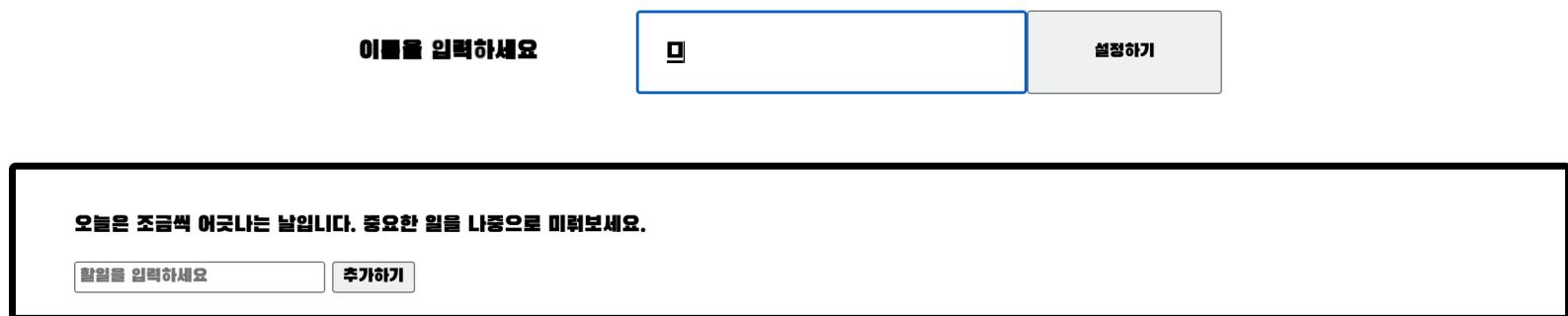
Local Storage를 활용하여 username 다루기

Local Storage를 활용하여 todo 목록 다루기

# Todo List 실습

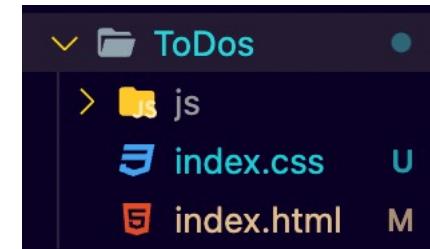
로컬 스토리지만으로 투두리스트를 구현해보자!

## 오늘의 목표



### Setup

새로운 폴더 생성 후,  
index.html 파일 /  
index.css 파일 /  
js 폴더를 만들어주세요!



### Setup

노션페이지에서  
css코드를  
index.css파일에  
복사 붙여넣기 해주세요!

# Todo List 실습1

Local Storage를 활용하여 username 다루기

**TASK1:** index.html파일을 생성한 후, 노션 페이지에서 코드를 복붙해주세요!

간단한 코드해설

Username 다루는 부분

Username 입력 받으면 활성화할 부분

새로고침할 때마다  
운세 나오는 부분

TodoList 다루는 부분

```
<body>
  <div class="usernameWrapper">
    <h3>이름을 입력하세요</h3>
    <input class="username"></input>
    <button style="width: 10%" type="button" onclick="setUsername()">설정하기</button>
  </div>
  <div id="header"></div>
  <div class="container">
    <div id="luck">
      <p></p>
    </div>
    <form id="todo-form">
      <input type="text" name="content" id="content" placeholder="할일을 입력하세요" required>

      <button class="submitBtn">추가하기</button>
    </form>
    <ul id="todo-list"></ul>
  </div>
  <script src=".js/localstorage.js"></script>
  <script src=".js/todo.js"></script>
  <script src=".js/luck.js"></script>
</body>
```

# Todo List 실습1

Local Storage를 활용하여 username 다루기

## 간단한 코드해설

### Math

`Math` 는 수학적인 상수와 함수를 위한 속성과 메서드를 가진 내장 객체입니다. 함수 객체가 아닙니다.

`Math` 는 `Number` 자료형만 지원하며 `BigInt` 와는 사용할 수 없습니다.

### Math.floor()

숫자와 같거나 작은 정수 중 가장 큰 수 반환

### Math.ceil()

숫자보다 크거나 같은 숫자 중 가장 작은 정수 반환

### Math.round()

소수점 반올림한 수와 가장 가까운 정수값을 반환

### Math.random()

0에서 1미만의 구간에서 랜덤하게 부동소수점 반환

**TASK2:** js폴더에 luck.js파일을 생성한 후 노션에서 코드를 복붙해주세요!  
( 운세도 확인해주세요 ^^ )

### luck.js

### 오늘의 운세 랜덤하게 나오는 부분

```
const lucks = [  
    "오늘은 운명같은 사람을 만날 수 있는 날입니다. 마음의 준비를 해주세요.",  
    "오늘은 코딩하기에 머리가 돌아가지 않는 날입니다. 조금 쉬세요.",  
    "오늘은 누군가 당신에게 야식을 사주는 날입니다. 메뉴를 생각해주세요.",  
    "오늘은 모든 일이 술술 풀리는 날입니다. 새로운 도전을 해보세요.",  
    "오늘은 조금씩 어긋나는 날입니다. 중요한 일을 나중으로 미뤄보세요.",  
    "오늘은 주량이 일시적으로 늘어난 날입니다. 술배틀에 참여해보세요.",  
];
```

```
const luck = document.querySelector("#luck p");  
const luckToday = lucks[Math.floor(Math.random() * lucks.length)];  
luck.innerText = luckToday;
```

0~5 중 랜덤한 숫자를 luckToday에 담는 역할

# Todo List 실습1

Local Storage를 활용하여 username 다루기

**TASK3:** username을 입력했을 때, 해당 값을 로컬 스토리지에 담아주기  
( js폴더에 localstroage.js 파일을 생성해주세요 )

## 간단한 코드해설

**username**

Input 요소를 담는 변수

**usernameWrapper**

Username다루는 div 요소를 담는 변수  
(앞의 우리의 실습 목표에서 봤듯이,  
이름을 입력하면 보이지 않아야 함)

**Header**

아직 보이지 않지만, 이를 입력을 완료하면  
활성화 되어야 하는 부분

**localStorage.js**

Dom요소에 접근하기

```
const username = document.querySelector(".username");
const usernameWrapper = document.querySelector(".usernameWrapper");
const header = document.querySelector("#header");
```

**header**

이름을 입력하세요

설정하기

**username**

오늘은 주량이 일시적으로 늘어난 날입니다. 술배틀에 참여해보세요.

할일을 입력하세요

추가하기

# Todo List 실습1

Local Storage를 활용하여 username 다루기

♥ 힌트 ♥

.value 를 통해서  
form요소의 값을 가져올 수 있다

로컬 스토리지의 새로운 값을 추가할 때  
다음과 같이 작성할 수 있다

```
window.localStorage.setItem(key, value)
```

**TASK3: username을 입력했을 때, 해당 값을 로컬 스토리지에 담아주기**

localStorage.js

```
const username = document.querySelector(".username");
const usernameWrapper = document.querySelector(".usernameWrapper");
const header = document.querySelector("#header");

/* 입력한 값으로 사용자의 이름을 설정하는 함수 */
// input의 값을 읽어오자!
// 해당값을 로컬 스토리지에 저장해보자 'username'이라는 키값으로 저장해보자

function setUsername() {
```

}

# Todo List 실습1

Local Storage를 활용하여 username 다루기

TASK3: username을 입력했을 때, 해당 값을 로컬 스토리지에 담아주기)

## 간단한 코드해설

[1] input요소를 담아두었던  
username으로부터 value를 통해 값을 읽어온다

[2] setItem을 통해 key는 'username'으로,  
value는 name변수에 담은 값으로 설정해서  
로컬 스토리지에 추가한다

[3] username.value에 빈 문자열을 할당하여  
비워준다.

## localStorage.js

```
/* 입력한 값으로 사용자의 이름을 설정하는 함수 */
function setUsername() {
    const name = username.value;
    window.localStorage.setItem("username", name);
    username.value = "";
}
```



# Todo List 실습1

Local Storage를 활용하여 username 다루기

**TASK4:** 입력한 username값을 바탕으로 다음과 같이 만들어주어야 한다  
(로컬스토리지에서 값을 읽어오기)

header

문자수 의 Todo List

초기화

입력 완료했을 때  
header 보이기

오늘은 운명같은 사람을 만날 수 있는 날입니다. 마음의 준비를 해두세요.

할일을 입력하세요

추가하기

userWrapper

이름을 입력하세요

설정하기

입력 받아야할 때  
userWrapper 보이기

오늘은 운명같은 사람을 만날 수 있는 날입니다. 마음의 준비를 해두세요.

할일을 입력하세요

추가하기



# Todo List 실습1

Local Storage를 활용하여 username 다루기

**TASK4:** 입력한 username값을 바탕으로 다음과 같이 만들어주어야 한다  
(로컬스토리지에서 값을 읽어오기)

Local Storage의 아이템 읽기

```
window.localStorage.getItem(key);
```

localStorage.js

```
/* 입력했던 사용자의 이름을 읽어오는 함수 */
// 로컬 스토리지에 저장했던 'username'을 읽어오자
// username이 있다면 usernameWrapper (입력받는 부분을 비활성화 하고)
// 활성화시킬 header부분에 element를 추가해주자

function checkUsername() {
    const checkName = window.localStorage.getItem("username");
    if (checkName) {
        입력 완료했을 때  
header 보이기
    } else {
        입력 받아야할 때  
userWrapper 보이기
    }
}
```



# Todo List 실습1

Local Storage를 활용하여 username 다루기

## 간단한 코드해설

### 템플릿 리터럴 Template literal

작은 따옴표('') 또는 큰 따옴표("") 대신  
백틱(``)을 사용한 문자열 표기법

① JS

```
1 const num1 = 10;
2 const num2 = 20;
3 console.log(`num1 + num2 = ${num1+num2}입니다.`);
```

## localStorage.js

```
/* 입력했던 사용자의 이름을 읽어오는 함수 */
function checkUsername() {
  const checkName = window.localStorage.getItem("username");
  if (checkName) {
    usernameWrapper.style.display = "none";
    header.innerHTML = `<h1> ${window.localStorage.getItem(
      "username"
    )} 의 Todo List</h1><button type="button" onclick="resetUsername()">초기화</button>`;
  } else {
    usernameWrapper.style.display = "flex";
    header.innerHTML = "";
  }
}
```

입력 완료했을 때  
header 보이기  
userWrapper 숨기기

입력 받아야 할 때  
userWrapper 보이기  
Header 숨기기



# Todo List 실습1

Local Storage를 활용하여 username 다루기

## localStorage.js

```
const username = document.querySelector(".username");
const usernameWrapper = document.querySelector(".usernameWrapper");
const header = document.querySelector("#header");

/* 입력했던 사용자의 이름을 읽어오는 함수 */
// 로컬 스토리지에 저장했던 'username'을 읽어오자
// username이 있다면 usernameWrapper (입력받는 부분을 비활성화 하고)
// 활성화시킬 header부분에 element를 추가해주자
function checkUsername() {
    const checkName = window.localStorage.getItem("username");
    if (checkName) {
        usernameWrapper.style.display = "none";
        header.innerHTML = `<h1> ${window.localStorage.getItem(
            "username"
        )} 의 Todo List</h1><button type="button" onclick="resetUsername()">초기화</button>`;
    } else {
        usernameWrapper.style.display = "flex";
        header.innerHTML = "";
    }
}
```

username 처리하는 로직 계속 유지

checkUsername();

```
/* 입력한 값으로 사용자의 이름을 설정하는 함수 */
// input의 값을 읽어오자!
// 해당값을 로컬 스토리지에 저장해보자 'username'이라는 키값으로 저장해보자
function setUsername() {
    const name = username.value;
    window.localStorage.setItem("username", name);
    username.value = "";
    checkUsername();
}
```

Username 처음 설정할 때도 해당 로직 실행

NEXT X LIKELION



# Todo List 실습1

Local Storage를 활용하여 username 다루기

**TASK5:** 초기화 버튼 클릭 시, 로컬 스토리지에 있는 username 삭제하고  
입력 받는 부분을 다시 활성화하기

## 간단한 코드해설

Local Storage의 아이템 삭제

```
window.localStorage.removeItem(key);
```

삭제할 때도 username로직을 처리하는  
checkUsername()을 실행시킨다.

## localStorage.js

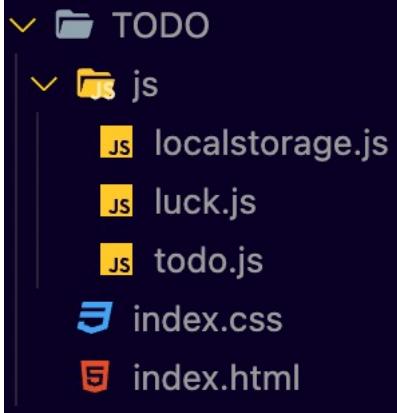
```
/* 입력했던 사용자의 이름을 초기화하는 함수 */
function resetUsername() {
    window.localStorage.removeItem("username");
    console.log("username 초기화");
    checkUsername();
}
```

# Todo List 실습2

Local Storage를 활용하여 todolist 다루기

## Setup

js 폴더에  
todo.js 파일 생성하기



Todo 항목 입력하는 부분

항목 추가하는 버튼

Todo 항목 나열될 목록

```
<body>
  <div class="usernameWrapper">
    <h3>이름을 입력하세요</h3>
    <input class="username"></input>
    <button style="width: 10%" type="button" onclick="setUsername()">설정하기</button>
  </div>
  <div id="header"></div>
  <div class="container">
    <div id="luck">
      <p></p>
    </div>
    <form id="todo-form">
      <input type="text" name="content" id="content" placeholder="할일을 입력하세요" required>
    </form>
    <button class="submitBtn">추가하기</button>
  </div>
  <ul id="todo-list"></ul>
</body>
```

# Todo List 실습2

Local Storage를 활용하여 todolist 다루기

Task1  
우리가 다룰  
DOM요소 담아두기

todo.js

```
/* Dom 요소 담아두기 */  
  
const todoForm = document.getElementById("todo-form");  
const todoList = document.getElementById("todo-list");  
const submitBtn = document.querySelector(".submitBtn");
```

입력 받을 부분

목록 다룰 부분

항목 추가 버튼

# Todo List 실습2

Local Storage를 활용하여 todolist 다루기

## 심화지식

JSON.stringify()  
JSON.parse()

### JSON.stringify()

JavaScript 값이나 객체를 JSON 문자열로 변환

```
console.log(JSON.stringify({ x: 5, y: 6 }));
// Expected output: '{"x":5,"y":6}'
```



### JSON.parse()

JSON 문자열의 구문을 분석하고, 그 결과에서 JavaScript 값이나 객체를 생성

```
const json = '{"result":true, "count":42}';
const obj = JSON.parse(json);

console.log(obj.count);
// Expected output: 42
```

# Todo List 실습2

Local Storage를 활용하여 todolist 다루기

심화지식  
filter함수

Array.filter()

주어진 함수의 테스트를 통과하는 모든 요소를 모아 새로운 배열로 반환

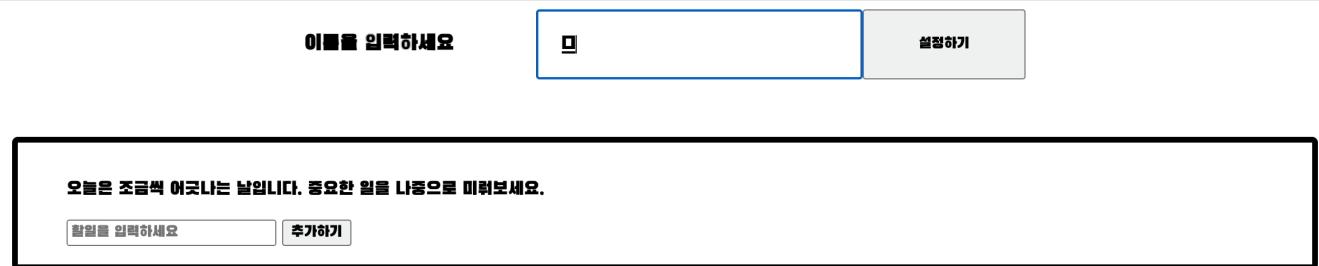
```
const words = ['spray', 'limit', 'elite', 'exuberant', 'destruction', 'present'];
const result = words.filter(word => word.length > 6);
```

word

result

# 갑자기 과제

Todo List 완성하기



## [단계별 가이드에 따라 TodoList완성하기]

- 새로고침했을 때도 todo목록이 그대로 유지
- Todo항목 삭제했을 때 로컬 스토리지에서  
도 해당 항목이 삭제되어있어야 함
- Todo항목 삭제 버튼을 눌렀을 때, 해당 항  
목만 적절하게 삭제되어야 함
- 로컬스토리지에 ‘배열’형태의 문자열로 저장  
하여 다를 것
- 심화개념학습에서 배웠던 것들을 활용하기