WEB프로젝트

이벤트 처리로 CSS 수정

prompt, alert 같은 브라우저 자체 기능이 아닌

자바스크립트로 직접 구현한 간단한 모달 창을 만들어본다.

해당 모달 창은 기본적으로 display: none 속성이 적용되어
사용자에게 보이지 않는 상태로 존재한다.

사용자가 열기 버튼을 클릭하는 이벤트가 발생하면
자바스크립트에서 display 속성을 수정하여
사용자에게 숨겨져 있던 모달 창을 보여주는 기능이다.

이를 통해 자바스크립트로 HTML의 요소 뿐만 아니라
요소에 적용되는 스타일도 수정하는 방법을 배울 수 있다.

index.html

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Modal</title>
    <meta charset="UTF-8" />
    <link rel="stylesheet" href="style.css">
  </head>

  <body>
    <h1>모달 창 TEST</h1>
    <p>열기 버튼을 누르면 숨겨져 있던 모달 창이 나타납니다</p>
    <button id="open">열기</button>
    <div class="modal-wrapper" style="display: none;">
      <div class="modal">
        <div class="modal-title">안녕하세요</div>
        <p>자바스크립트로 display속성을 수정하여 모달 창을 나타냅니다</p>
        <div class="close-wrapper">
          <button id="close">닫기</button>
        </div>
      </div>
    </div>
    <script src="index.js"></script>
  </body>
</html>
```

index.js

```
const open = document.getElementById("open");
const close = document.getElementById("close");
const modal = document.querySelector(".modal-wrapper");

open.onclick = () => {
    modal.style.display = "flex";
};
close.onclick = () => {
    modal.style.display = "none";
};
```

숫자 카운터와 동일하게 Id 선택자 or 쿼리셀렉터를 이용하여 HTML 요소를 가져온다.

마찬가지로 동일하게 해당 요소에 onclick 이벤트가 발생할 시 modal 요소의 display 속성을 none 에서 flex로 변경하여 사용자에게 모달 창을 보여준다.

그 후 닫기 버튼을 누르면 다시 display 속성을 none으로 변경하여 창을 숨긴다.

이미지 슬라이더

addEventListener와 자연스러운 이미지 전환을 위한
애니메이션이 적용된 이미지 슬라이더 페이지를 구현한다.

index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>이미지 슬라이드</title>
  <link rel="stylesheet" href="styles.css" />
</head>

<body>
  <div class="slider-container">
    <div class="slide">
      <img src="./img/img1.jpg" class="slide-img" alt="" />
      <h1>1</h1>
    </div>
    <div class="slide">
      <h1>2</h1>
    </div>
    <div class="slide">
      <h1>3</h1>
    </div>
    <div class="slide">
      <h1>4</h1>
    </div>
  </div>
  <div class="btn-container">
    <button type="button" class="prevBtn">
      prev
    </button>
    <button type="button" class="nextBtn">
      next
    </button>
  </div>
  <script src="index.js"></script>
</body>
</html>
```

style.css

```css
*{
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

img{
  width: 100%;
  display: block;
}

.section {
  padding: 5rem 0;
}

.section-center {
  width: 90vw;
  margin: 0 auto;
  max-width: 1170px;
}

main {
  min-height: 100vh;
  display: grid;
  place-items: center;
}

.slider-container {
  border: 5px solid #ffffff;
  width: 80vw;
  margin: 0 auto;
  height: 40vh;
  max-width: 80rem;
  position: relative;
  border-radius: 10px;
  overflow: hidden;
  margin-top: 4rem;
}
```

```css
.slide {
  position: absolute;
  width: 100%;
  height: 100%;
  background: #cccccc;
  display: grid;
  place-items: center;
  transition: all 0.25s ease-in-out;
  text-align: center;
}
.slide-img {
  height: 100%;
  object-fit: cover;
}
.slide h1 {
  font-size: 5rem;
}
.person-img {
  border-radius: 50%;
  width: 6rem;
  height: 6rem;
  margin: 0 auto;
  margin-bottom: 1rem;
}
.slide:nth-child(1) h1 {
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
}

.slide:nth-child(2) h1 {
  color: #dddddd;
}
```

style.css

slide:nth-child – 슬라이드 클래스의
n번째 요소를 선택한다
(nth-child(2) : slide클래스의 두번째 요소를 선택)

background, border-color : transparent = 투명화

text-transform: capitalize = 단어의 첫 글자 대문자

```css
.slide:nth-child(2) {
  background: linear-gradient(rgba(0, 0, 0, 0.4), rgba(0, 0, 0, 0.4)),
  url("./img/img2.jpg") center/cover no-repeat;
}

.slide:nth-child(3) {
  background: url("./img/img3.jpg") center/cover no-repeat;
}

.slide:nth-child(4) {
  background: url("./img/img4.jpg") center/cover no-repeat;
}

.btn-container {
  display: flex;
  justify-content: center;
  margin-top: 0.75rem;
}
.prevBtn,
.nextBtn {
  background: transparent;
  border-color: transparent;
  font-size: 1.75rem;
  cursor: pointer;
  margin: 0 0.25rem;
  text-transform: capitalize;
  letter-spacing: 2px;
  color: #1f1f1f;
  transition: all 0.3s linear;
}
.prevBtn:hover,
.nextBtn:hover {
  color: rgb(216, 216, 216);
}
```

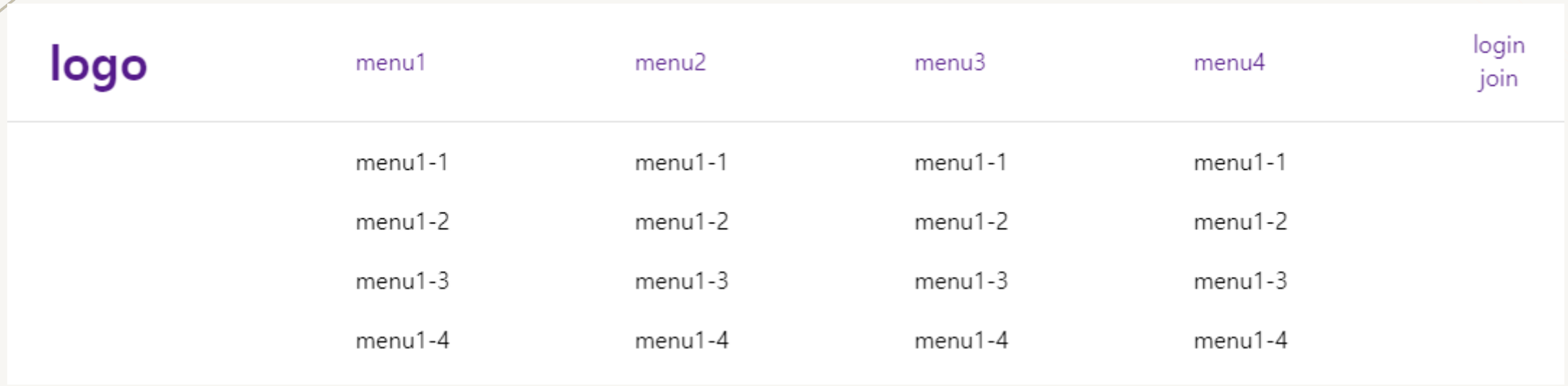
index.js

next, prev 버튼을 누를 때 마다 발생하는 이벤트를 처리한다.

현재 이미지의 순번을 counter 변수에 저장하고,
counter 값에 따라 이미지 슬라이드를 이동,
next, prev 버튼을 감추거나, 표시하는 기능을 구현한다.

```
const slides = document.querySelectorAll(".slide");
const nextBtn = document.querySelector(".nextBtn");
const prevBtn = document.querySelector(".prevBtn");
slides.forEach(function (slide, index) {
  slide.style.left = `${index * 100}%`;
});
let counter = 0;
nextBtn.addEventListener("click", function () {
  counter++;
  imgslide();
});

prevBtn.addEventListener("click", function () {
  counter--;
  imgslide();
});

function imgslide() {
  if (counter < slides.length - 1) {
    nextBtn.style.display = "block";
  } else {
    nextBtn.style.display = "none";
  }
  if (counter > 0) {
    prevBtn.style.display = "block";
  } else {
    prevBtn.style.display = "none";
  }
  slides.forEach(function (slide) {
    slide.style.transform = `translateX(-${counter * 100}%)`;
  });
}

prevBtn.style.display = "none";
```

네비게이션 바



수평으로 된 네비게이션 바를 구현한다.

메뉴에 마우스를 올리는 이벤트가 발생할 경우, 자바스크립트로 하위메뉴에 display 속성을 추가하여 화면에 나타나게 한다.

css의 ::before, ::after 가상요소를 사용

index.html

네비게이션 메뉴를 만들기 위한 html 코드를 작성한다.

div class=snb : 클릭 시 나오는 하위 메뉴

각각 생략된 코드는 menu1과 동일한 내용을 입력한다.

```html
<div id="wrap">
        <div id="header">
            <h1>
                <a href="#">logo</a>
            </h1>
            <ul id="gnb" class="">
                <li>
                    <a href="#">menu1</a>
                    <div class="snb">
                        <ul>
                            <li>menu1-1</li>
                            <li>menu1-2</li>
                            <li>menu1-3</li>
                            <li>menu1-4</li>
                        </ul>
                    </div>
                </li>
                <li>
                    <a href="#">menu2</a>
                    <!-- 중간 코드 생략 -->
                </li>
                <li>
                    <a href="#">menu3</a>
                    <!-- 중간 코드 생략 -->
                </li>
                <li>
                    <a href="#">menu4</a>
                    <!-- 중간 코드 생략 -->
                </li>
            </ul>
            <ul id="util">
                <li><a href="#">login</a></li>
                <li><a href="#">join</a></li>
            </ul>
        </div>
    </div>
```

style.css

*로 전체 스타일의 margin, paddin을 초기화.

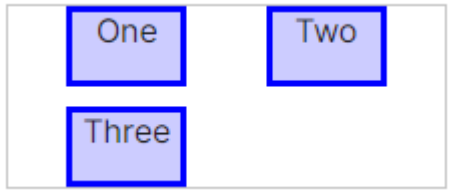
justify-content : flex 레이아웃에서 요소를 정렬한다.

space-between

첫 아이템은 시작지점에 마지막 아이템은 끝지점에 배치한다

space-evenly

시작, 중간, 끝 모든 간격을 동일하게 만든다.



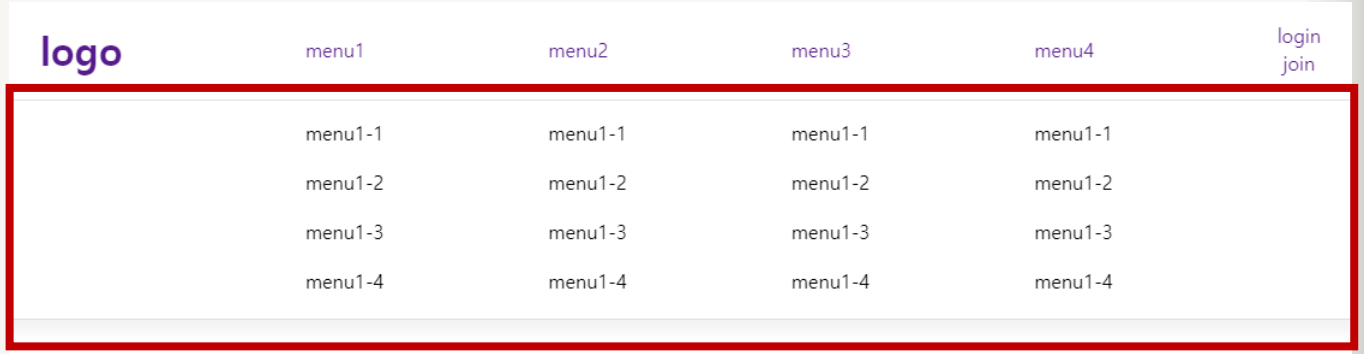

```css
*{
    margin: 0;
    padding: 0;
}

ul,li{
    list-style: none;
}

a{
    text-decoration: none;
}

#wrap{
    width: 100%;
}

#header{
    position:fixed;
    width: 100%;
    left: 0;
    top: 0;
    padding: 0 30px;
    box-sizing: border-box;
    border-bottom: 1px solid #e1e1e1;
    display:flex;
    justify-content: space-between;
}

#header h1{
    padding: 20px 0;
}

#header > #gnb {
    width: 1200px;
    display: flex;
    justify-content: space-evenly;
    align-items: center;
}
```

::before – html에 실제로는 존재하지 않는 가상의 요소이며

before는 선택된 요소의 첫 번째 자식 요소,

after는 선택된 요소의 마지막 자식 요소로 가상의 요소를 생성한다.

실제 DOM에 존재하지 않지만, CSS로 스타일을 지정할 수 있으며,
항상 content 속성이 포함되어야 하며, 이를 통해 가상 요소의 내용을
추가하며, 주로 아이콘, 구분선, 버튼, 말풍선 등을 추가하는데 활용된다.



하위 메뉴가 표시될 공간의 스타일을 적용하는데 사용됨.

```css
#header > #gnb::before {
    content: "";
    width: 100%;
    position: absolute;
    top: 85px;
    left: 0;
    height: 180px;
    background: #fff;
    border-bottom: 1px solid #e1e1e1;
    box-shadow: 0px 10px 10px 0px rgba(48,49,51,6%);
    display: block;
}

#header > #gnb::before {
    display: none;
}

#header > #gnb.on::before {
    display: block;
}

#header > #gnb > li .snb {
    position:absolute;
    top: 100px;
    display:none;
}

#header > #gnb.on > li .snb {
    display: block;
}

#header > #gnb > li > .snb > ul > li + li {
    margin-top:20px;
}

#util {
    display: flex;
    flex-direction: column;
    justify-content: center;
    text-align: center;
    line-height: 1.4;
}
```

index.js

mouseover 이벤트가 발생한 경우,
#gnb요소에 on이라는 클래스를 추가한다.

on이라는 클래스가 추가되는 순간
css에 미리 작성되어 있던 on클래스에 대한
스타일이 적용이 되며, 하위메뉴가 표시된다.

그 후, #gnd와 하위메뉴를 모두 포함하는
#header 범위에서 mouseout이 될 경우,
추가되었던 on 클래스가 삭제되고
나타났던 하위메뉴가 다시 사라지게 된다.

```
let gnb = document.querySelectorAll("#gnb > li")
let gnbElement = document.querySelector("#gnb")

for (let i = 0; i < gnb.length; i++) {
    gnb[i].addEventListener("mouseover", () => {
        gnbElement.classList.add("on")
    })
}

let headerElement = document.querySelector("#header")

header.addEventListener("mouseout", (e) => {
    if (e.target.id == "gnb") {
        gnbElement.classList.remove("on")
    }
})
```

종합 페이지 구현

네비게이션

모달

이미지 슬라이드

스크롤 스냅

기타

반응형 웹(미디어 쿼리)

웹 폰트

지금까지 구현했던 웹페이지의 다양한 기능들을 모아서 하나의 페이지를 만들어본다.

(코드 재사용 가능)