



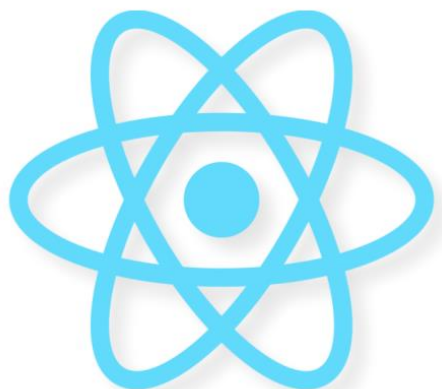
LEOPOLD

사이트 모바일 + 웹 가이드문서

작성자 안주현
작성일 2024.01.09

사용 기술

REACT



jQuery



<HTML>
HTML



{CSS}
CSS



(JS)
JavaScript



제작 의도

개발 목적 : 구매 및 판매

메인 페이지 디자인 개선

구현 목표 : 편리한 제품 선택

- 제품 디자인 필터 기능
- 용도에 따른 검색기능

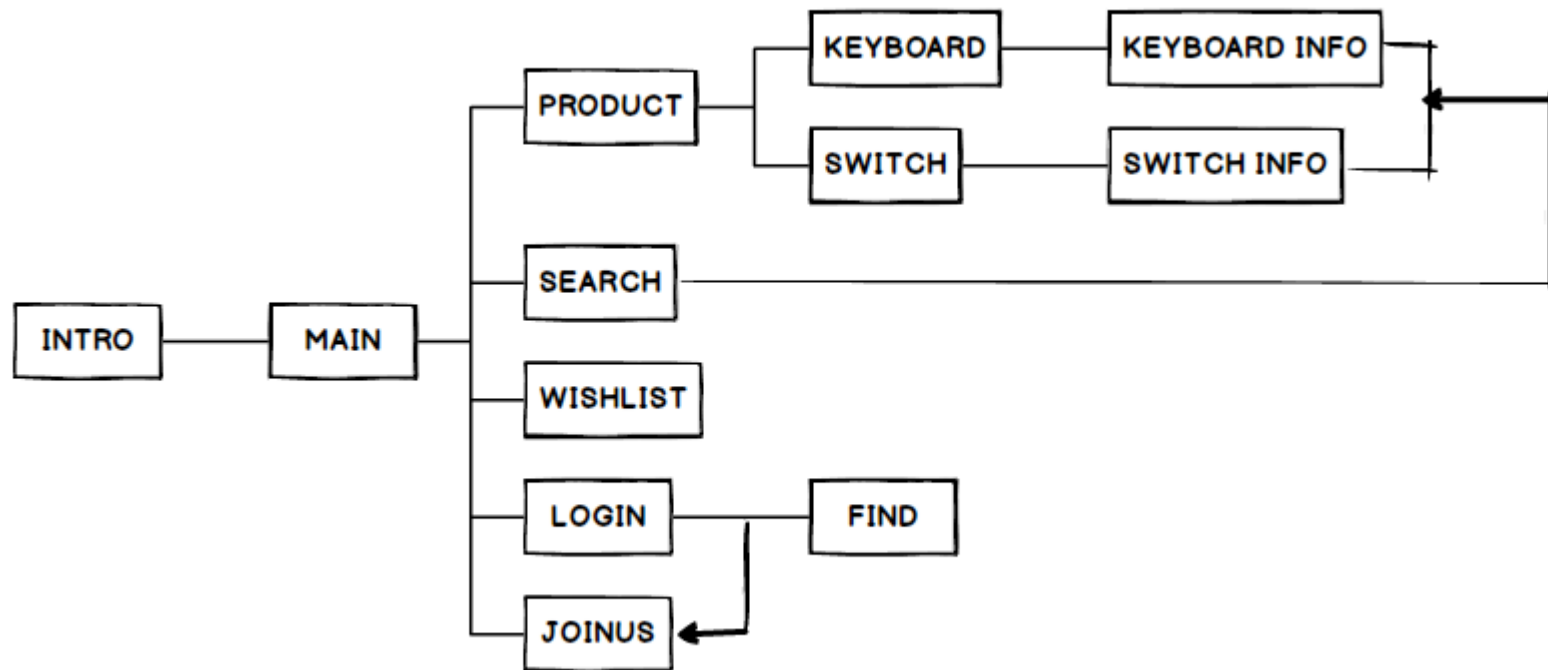
구현 목표 : 편리한 정보습득

Toggle 을 이용한 빠른 페이지 전환

추가 개선 : Menu 컨셉

키보드 사이트의 컨셉을
활용 한 타이핑 효과 적용

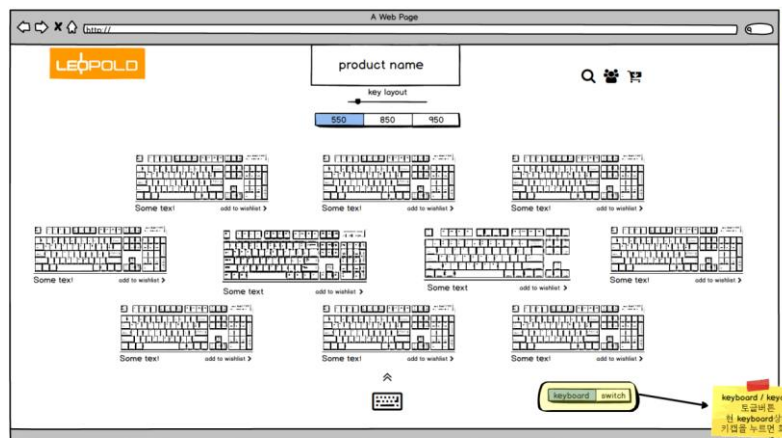
분석 설계 - 사이트 맵



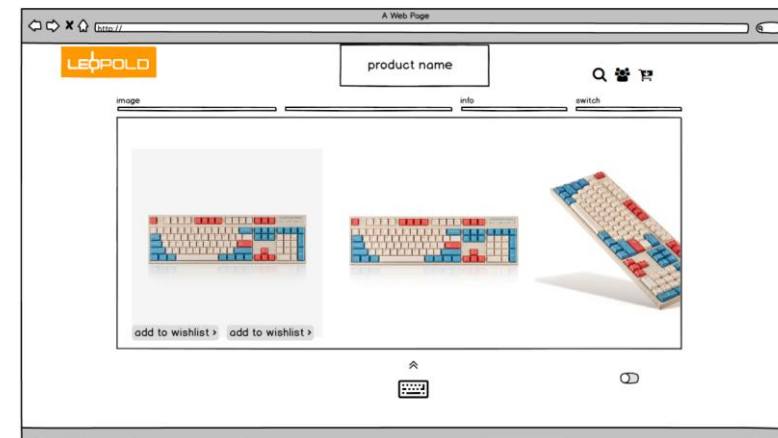
분석 설계 - 웹 페이지



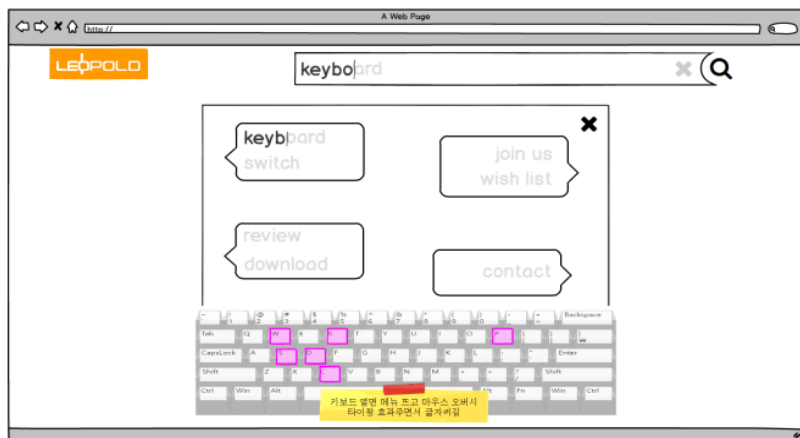
인트로 페이지



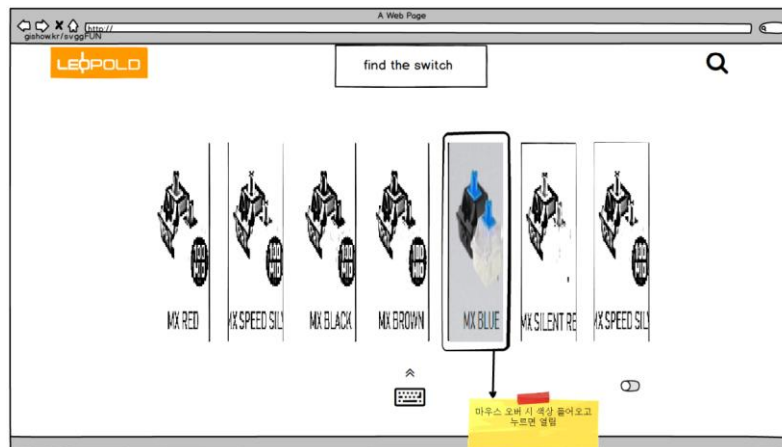
메인 페이지



메인 상세 페이지



메뉴 페이지



스위치 페이지



스위치 상세 페이지

분석 설계 - 웹 페이지

A Web Page

LEOPOLD

rog in

* user id

* password

Forgot your id?
Forgot your password?
Don't have an account?

rogin

선택사항 제외
필수등의 체크하면
동의사항 체크가 안되면
모달창으로 체크하라 안내문이 좋음

로그인 페이지

A Web Page

LEOPOLD

join us

id

pw

pw check

name

e-mail

rogin

ajax로 blur일 때
유효성 검사
아이디는 중복검사를 실행
해야 함
pw는 아이디와 일치 불가
name은 아이디 pw와 일
치 불가
email은 인증

회원가입 페이지

A Web Page

LEOPOLD

search

상품 : 00건

이미지

상품정보

제품가격

검색결과 더보기

정보 : 00건

이미지

정보

종류

검색결과 더보기

검색 페이지

A Web Page

LEOPOLD

find id / pw

< forgot id forgot pw >

name

e-mail

id

pw

find

ajax로 blur일 때
입력한 정보와 일치하는
정보가 없다면
입력한 아래에 빨간 줄
있다면 그대로 넘어감
find 위한 완벽하게 일치하
는 정보만 보내줌
pw찾기는 초기화 된
비밀번호를 화면에 노출

회원찾기 페이지

A Web Page

LEOPOLD

wishList

선택	이미지	상품정보	수량	단가	합금액	재용비	선택
<input checked="" type="checkbox"/>		상품명, 가격, 옵션, 색상, 브랜드, 카테고리	Combobox	₩1,000	₩1,000	₩1,000	Order Delete
<input checked="" type="checkbox"/>		상품명, 가격, 옵션, 색상, 브랜드, 카테고리	Combobox	₩1,000	₩1,000	₩1,000	Order Delete

총 상품금액 00,000,000원 + 총 배송비 00,000,000원 = 결제 예정금액 00,000,000원

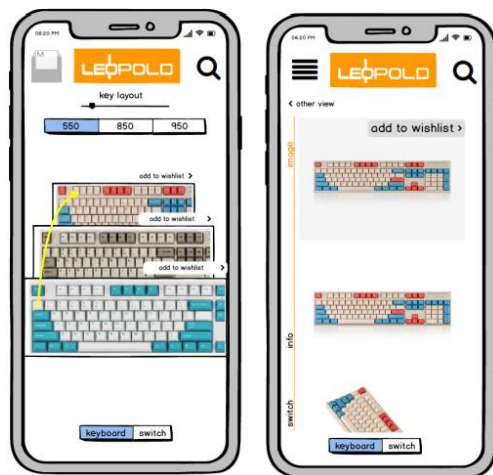
All List Order Selected List Order Selected Delete

장바구니 페이지

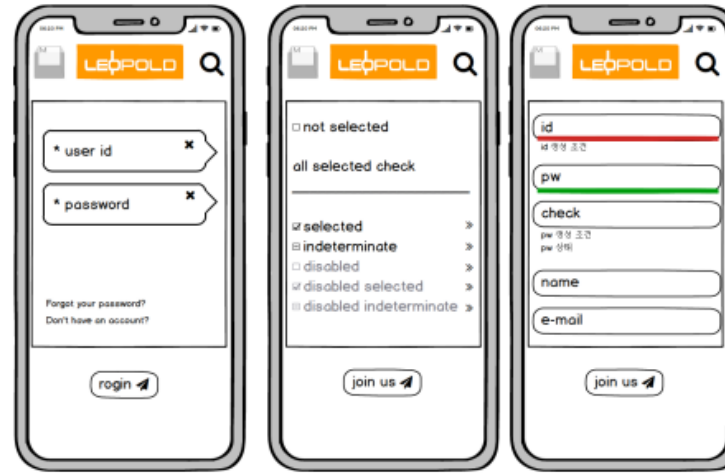
분석 설계 - 모바일



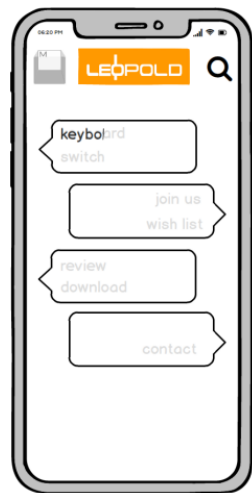
인트로 페이지



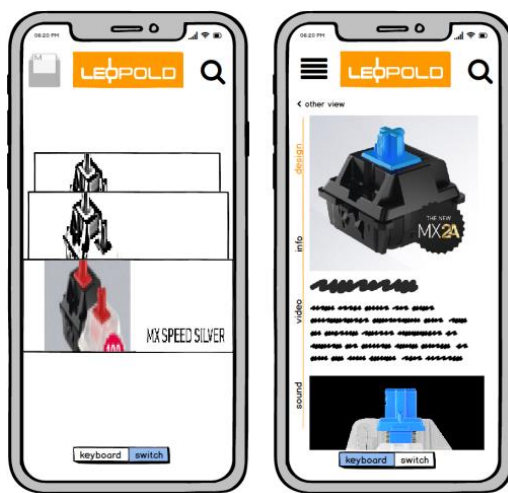
메인 페이지



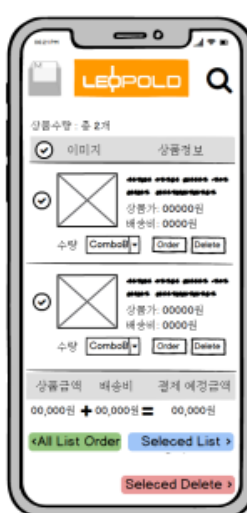
로그인 페이지



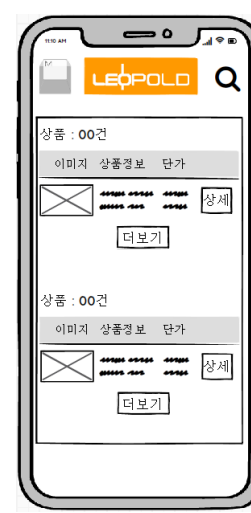
메뉴 페이지



스위치 페이지

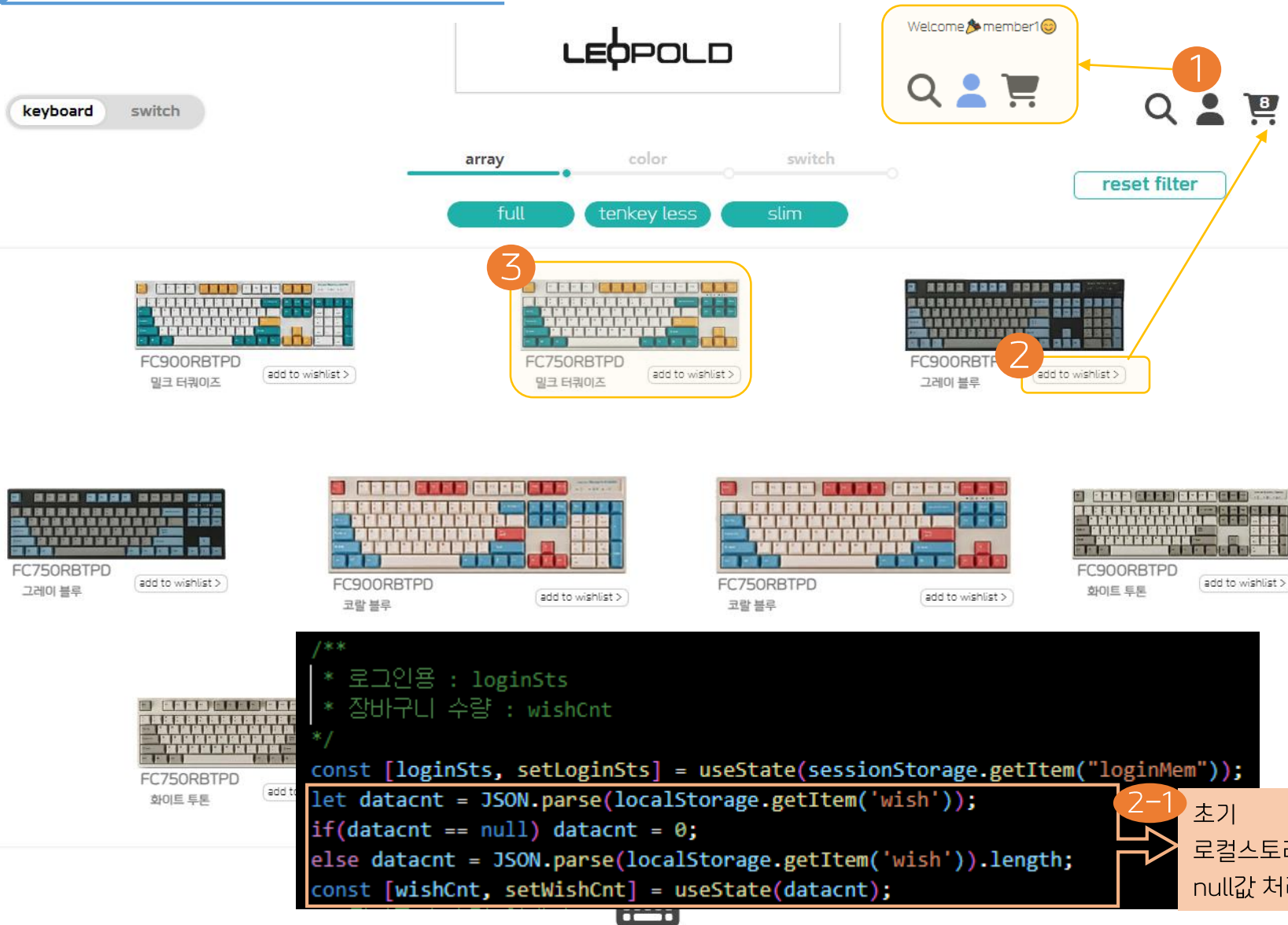


장바구니 페이지



검색 페이지

페이지 구현 메인-로그인/장바구니



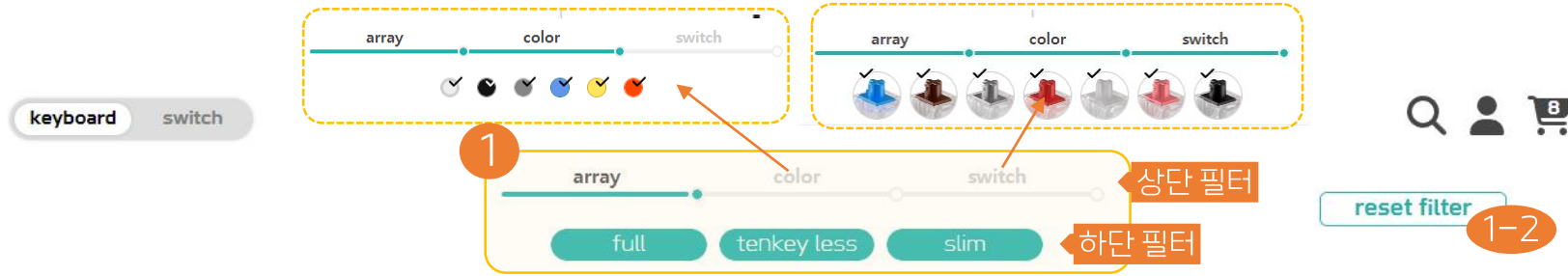
- 1 세션스토리지에 로그인 정보 저장
로그인 상태 관리를 위해 **useState** 사용
로그인 상태
 - 아이콘 변경
 - 상단에 로그인 아이디 정보
 - 로그인페이지 / 회원가입 페이지 접근 불가**로그아웃 시**
 - 로그인 페이지 접근 가능

- 2 로컬스토리지에 위시리스트 저장
장바구니 수 증가를 위해 **useState** 사용
신규 상품 클릭) 장바구니 수 증가
중복 상품 클릭) 제품 수량 증가
장바구니 수량은 제품 종류만 관련 됨
카트 아이콘을 통해 장바구니 정보 접근
초기 로컬스토리지 null값 처리 **2-1**

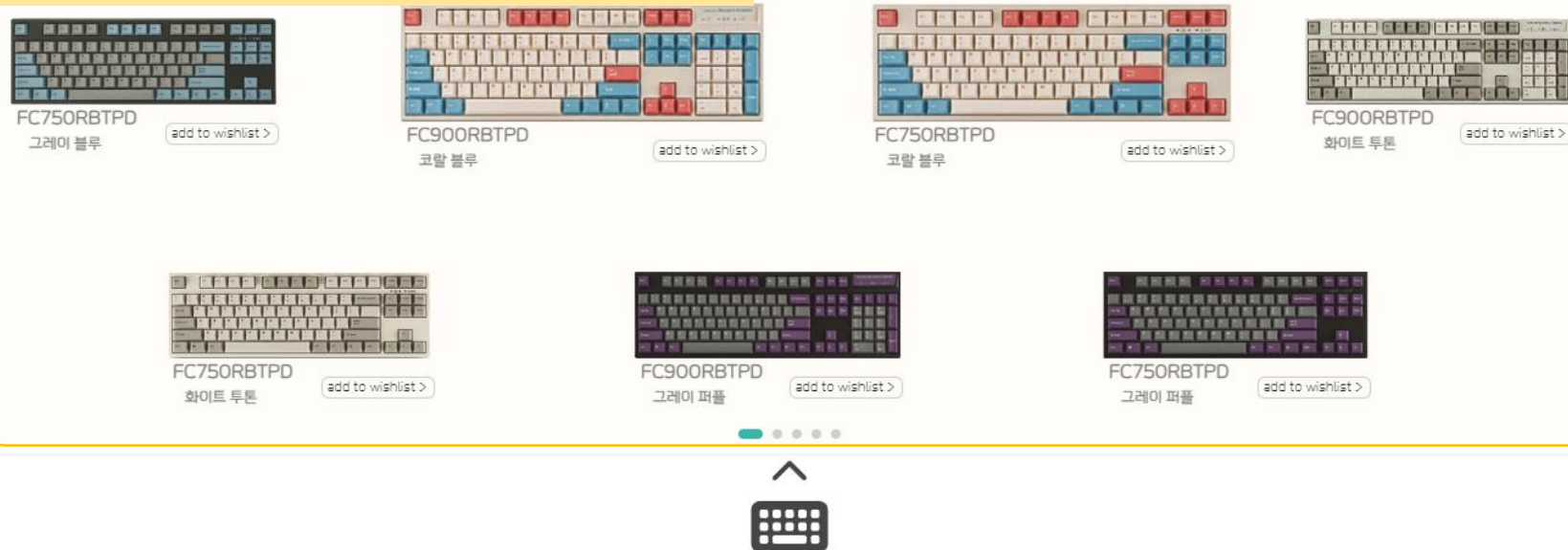
- 3 제품 상세페이지 이동
useNavigate 사용 state로 제품 코드 전달
클릭 상품의 상세보기 페이지 이동

2-1 초기
로컬스토리지
null값 처리

페이지 구현 메인-필터



- 2 제품 영역
- 원 페이지 구성을 위해 **Swiper plugin** 적용
 - 드래그 또는 키보드로 제품 목록 이동
 - 하단 **pagination** 표시
 - 페이지 당 최대 10개의 상품 노출



- 1 상/하단 다중 필터
- 상단 선택 시 하단 필터 변경
- 상단은 랜더링을 위해 **useState**
- 상단 필터 변경 시 상단만 변경되도록 제품 영역은 **memo**처리
- 하단 필터는 옵션값을 **useRef**로 관리
- 제품 영역에 props로 전달하고 해당 데이터를 **useState**를 이용해서 관리
 - 페이지 변경과 필터 변경을 구분하는 **optionChangeState** 변수를 **useRef**로 관리
 - 필터 초기화를 위한 강제 리랜더링 변수 **useState**로 관리 1-2

1

```
// 강제 리랜더링
const [force, setForce] = useState(null);
// 대분류/세부분류
const [optSel, setOptSel] = useState("array");
// 세부분류 옵션 -> 변경 시 데이터변수 변경
const arrayOpt = useRef(["full", "tenkey less"]);
const colorOpt = useRef(["co-wt", "co-bk", "co-br"]);
const switchOpt = useRef(["sw-bu", "sw-br", "sw-bl"]);
// 옵션 변경 상태
const optionChangeState = useRef(false);
// 데이터 변수
const dataIdx = useRef(prodList);
```

페이지 구현 메인-토글/검색(검색 페이지)

1

keyboard switch

LEOPOLD

2



Search

x

Filter by keyword

Search

2-1

2

검색 버튼

상단에서 검색 창 보여짐

2-1

검색 버튼 클릭 시 검색어가 없는 경우 경고 처리 후 검색 창에 커서 이동

- 텍스트가 있는 경우 검색페이지로 이동
- 키워드는 `useNavigate state`로 전달

2-2




검색 페이지

- 키워드로 상품검색 및 스위치 정보 검색
- View detail 버튼을 이용하여 각 제품의 상세페이지로 이동
- View more 버튼을 이용하여 리스트를 3개씩 추가, 이때 보여줄 리스트 수를 `useState`로 처리하여 페이지 리랜더링


2-2

Search Result

Product : 19 items

Image	Product Information	Price	View
	FC900RBTPD - 코랄 블루- full design keyboard	₩178,000	View detail ↗
	FC750RBTPD - 코랄 블루- tenkey less design keyboard	₩175,000	View detail ↗
	FC900RBTPD - 그레이 블루- full design keyboard	₩178,000	View detail ↗
View more...			

Switch : 1 items

Image	Switch Information
	click-blue(클릭 블루) : 경쾌한 클릭음과 리드미컬한 타이핑

```
/**
 * [ 상태관리 변수 ]
 * prodResultCount
 * switchResultCount
 */
const [prodCount, setProdCount] = useState(3);
const [switchCount, setSwitchCount] = useState(3);
```

페이지 구현 스위치 페이지

LEOPOLD



2

keyboard switch

1

linear
red

silent
silver

일반적인 압력의 키압과 고속 타이핑

view
more

linear
black

nonclick
brown

click
blue

silent
red

nonclick
clear

3

linear
red



부드럽고 걸림이 없는
키감과 조용한 타이핑

silent
silver



일반적인 압력의 키압
과 고속 타이핑

linear
black



부드럽고 걸림이 없는
키감과 조용한 타이핑

nonclick
brown



부드럽고 구분감 있는
키감과 조용한 타이핑

click
blue



경쾌한 클릭음과 리드
미컬한 타이핑

silent
red



소음을 30% 줄인 조
용하고 부드러운 타이
핑

nonclick
clear



다소 높은 키압력과
강한 구분감이 있는
타이핑

1

Flex display

마우스 오버 된 스위치를 크게 보여
주고 간단한 정보 제공

- 클릭 시 상세페이지 이동
- `useNavigate state`로 클릭 정
보 전달

2

메인 페이지 (keyboard) 이동용
토글 버튼

3

모바일에서 가독성을 위해
`Flex direction column`으로 변경



페이지 구현 장바구니 페이지

LEOPOLD

keyboard switch



WishList

1 SubTotal(5 items)

checked order

2 checked delete

2-1

<input checked="" type="checkbox"/>	Image	Product	Price	Quantity	Sub Total	Select
<input checked="" type="checkbox"/>		FC750RBTPD - 밀크 터퀴이즈	₩175,000	- 1 +	₩700,000	order delete
<input checked="" type="checkbox"/>		FC900RBTPD - 코랄 블루	₩178,000	- 4 +	₩178,000	order delete
<input checked="" type="checkbox"/>		FC750RBTPD - 그레이 퍼플	₩175,000	- 1 +	₩350,000	order delete
<input checked="" type="checkbox"/>		FC980MBTPD - 화이트 투톤	₩182,500	- 2 +	₩182,500	order delete
<input checked="" type="checkbox"/>		FC900RBTPD - 그레이 퍼플	₩178,000	- 1 +	₩178,000	order delete

₩1,588,500(Product Price) + ₩2,500(delivery fee) =

2-3 Total ₩1,591,000

delete confirm

Do you really want to delete it????

2-2

Delete

Cancel

장바구니

로컬스토리지에서 위시리스트 정보 불러오기

제품별 수량 업데이트

- 1 - 1개 이하 item 2개 이상 items
- 2 - 체크 항목 지우기 / 개별 항목 지우기
 - 제품 별 체크 된 항목만 삭제
 - 개별 delete클릭 시 해당 제품만 삭제
 - 체크박스는 전체 체크에만 상단 체크
 - 삭제하면 모달 창 띄우고 Delete 클릭 시에만 해당제품 삭제
 - 합계금액 업데이트

제품 수량 변경

- 3 - 로컬에 바로 적용
 - 합계와 하단 총 금액 업데이트



페이지 구현 상세 페이지

1

image1 image2 image3 must-read info1

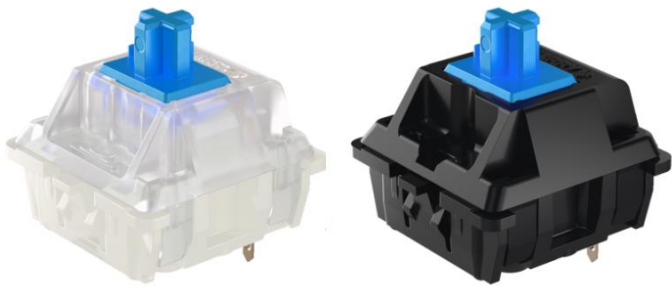
메인 상세 페이지

2



white-image black-image information motion

스위치 상세 페이지



스위치 정보

성향	클릭감이 좋고 경쾌함
특성	촉각 및 가청 스위칭 특성
키압	60cN 작동력
이동	2.2mm 사전 이동
범위	총 이동거리 4.0mm
소음	High
용도	입문자/타이핑

show related product>

1

네비게이션

- 클릭으로 내부 정보로 접근
- 하단 요소 크기에 따른 네비게이션 길이 동적 세팅
- 이미지 정보가 긴 경우 네비게이션을 이용하면 상단으로 바로 이동

2

```
// 이미지 세로크기 저장 함수
const sizeCheck = (ele) => {
  // console.log(ele);
  imgWidSize.length = 0;
  // 이동거리 저장용 임시변수
  let xpos = 0;
  // 순번값
  let seq = 0;
  // 이미지 세로크기 배열에 넣기
  ele.each((i, v) => {
    // 개별 이미지 길이 저장
    imgHeiSize[i] = v.height;
    // 가로 이동거리 저장
    xpos += v.width;
    // 이미지 별 이동값 저장(네비용)
    imgWidSize[i] = xpos - limitW <= 0 ? 0 : xpos - limitW;
    if (v.height > v.width) {
      // 가로영역 이동한계값, 세로 한계값, 움직일 요소
      pos[seq] = [xpos - limitW, v.height - limitH, infoImg.eq(i)];
      seq++;
      // console.log(pos);
    }
  });
  // 전체 이동거리 업데이트
  all = xpos - limitW;
  // 위치이동 포인트 설정(이미지 가로길이보다 세로길이가 크면 세로스크롤)
}; /////////////// 사이즈 저장 함수 ///////////////////
```

메인 정보

- 가로 스크롤 이동과 세로 스크롤 이동이 자동으로 조정 됨
- 내부 정보 크기에 따라서 이동할 방향의 한계 값, 움직일 요소를 자동으로 지정해 줌

페이지 구현 상세 페이지

스크롤 이벤트 적용

스크롤 이벤트 시 box의 기존 값인 position().left를 전달

event.wheelDelta를 읽어와서 휠 방향 처리

데이터를 적용하기 전에 스크롤 이벤트 함수를 적용하여 파라미터 값이 전달되지 않는 오류를 막기위해

Promise()메서드 처리

```
//x값 세팅 함수(promise)
const setX = (val) =>
  new Promise((resolve, reject) => {
    if (val !== null) {
      resolve(val);
    } else {
      reject(val);
    }
  });
```

```
// 휠 이벤트 주기
document.querySelector(".prod-info").addEventListener("wheel", (event) => {
  let delta = event.wheelDelta > 0 ? 1 : -1;
  setX(infoBox.position().left)
    .then((res) => {
      ////////// 광스크롤 막기 //////////
      if (psts === 1) return true; //돌아가!
      psts = 1; //잠금!
      setTimeout(function () {
        psts = 0; //해제
      }, 50); //0.05초후 해제//////////
      x = res;
      infoScroll(delta, x);
    })
    .catch((e) => {
      console.log(e);
    });
});
```


페이지 구현 상세 페이지

가로세로 이동 구현

- 데이터 height값을 저장한 데이터를 휠 방향에 따라 적용

가로 이동

- 휠 방향으로 분기
- 아래 휠: height position().top 값과 일치하지 않으면 가로 이동
- 위 방향 휠: height position().top != 0이면 가로 이동
- 반대는 세로 이동 함수 호출

세로 이동:

- 0부터 한계 값까지 y축 이동

```
// 기능 : 가로세로 스크롤을 조합하여 박스를 이동시킨다.
// 파라미터 (이미지 전체박스)
const infoScroll = (delta,x)=>{
  // 휠 이벤트 시 네비설정
  for(let i=0; i < imgWidSize.length; i++){
    if(x <= -imgWidSize[i]){
      addOnNav(i)
    }
  }
  if (x >= 0 && delta > 0) x = 0;
  else if (x >= -pos[0][0]) x = horizonScroll(pos[0], delta);
  else if (x >= -pos[1][0]) x = horizonScroll(pos[1], delta);
  else if (x >= 0 && delta <= 0) x = 0;
  if (x <= -pos[1][0] && delta < 0) x = -pos[1][0];
  infoBox.css("left", x + "px");
}; ////////////// infoScroll 함수 ///////////////////
```

```
// 가로스크롤 함수
function horizonScroll(target, dir) {
  // target - 이동대상 dir - 방향
  x += MOVE * dir;
  if (dir === -1) {
    //아래방향 스크롤
    if (x > -target[0]) y = 0; //y고정 x이동
    else if (x <= -target[0] && target[2].position().top != -target[1]) {
      //x고정 y이동
      x = -target[0];
      verticalScroll(target, dir);
    }
  } else {
    //윗방향 스크롤
    if (target[2].position().top != 0) {
      // x고정 y이동
      x -= MOVE * dir; // x축 이동 초기화
      verticalScroll(target, dir);
    }
  }
  return x;
} ////////////// 가로스크롤 함수 ///////////////////
```

```
// 기능 : 이미지의 세로높이를 받아와서 세로스크롤을 수행한다.
// 파라미터(이동대상, 이동값)
function verticalScroll(target, dir) {
  // target - 대상배열
  y += MOVE * dir;
  if (y > 0) y = 0;
  else if (y < -target[1]) y = -target[1];
  target[2].css("top", y + "px");
} ////////////// 세로스크롤 함수 ///////////////////
```


페이지 구현 메뉴 페이지

KEYBOARD

SWITCH

LOGIN

REGISTER

WISHLIST

CONTACT

메뉴 페이지

- 마우스 오버 시 키보드 눌리는 효과(CSS + JS Interval)
- 마우스 오버 시 상단의 타이핑 효과(CSS + JS Interval)
- 각 이벤트 전에 **clearInterval** 를 우선하여 이벤트 중복 제거
- 클릭으로 링크 이동



```
// 초마다 글자를 입력하는 인터발 함수
function insertText(txt, area) {
  // 데이터 초기화
  let count = 0;
  let inTxt = keylist[txt][0];
  let tglst = [];
  keylist[txt][1].forEach((ele,idx) => {
    tglst[idx] = $('.key').filter(`:eq(${ele})`);
  });
  clearInterval(autoI);
  // 인터발 함수
  autoI = setInterval(() => {
    $(area).append(inTxt[count]);
    tglst[count].addClass('push');
    count++;
    reset(tglst[count-1]);
    // 글자데이터 길이보다 길어지면 멈춤
    if (count >= inTxt.length) {
      clearInterval(autoI);
    }
  }, TYPING_TIME);
} // insertText 함수 //
```

JS

```
/* 키 입력 */
.key.push {
  animation: push-key-ani .2s linear;
}
.key.push .key-top aside{
  animation: push-key-ani2 .2s linear;
}
/* 키 입력 애니메이션 */
@keyframes push-key-ani {
  20%{
    transform: translateY(10px);
  }100%{
    transform: translateY(0px);
  }
}
@keyframes push-key-ani2 {
  20%{
    background-color: cornflowerblue;
  }100%{
    background-color: #fff;
  }
}
```

CSS

```
// 스타일 초기화 함수
function reset(ele) {
  // 리셋 setTime 함수 호출
  setTimeout(() => {
    ele.removeClass("push");
  }, 200);
} // reset 함수 //
```

JS

```
// 초기화 함수
function resetAutoI() {
  clearInterval(autoI);
}
```

JS

페이지 구현 메뉴 페이지

1



```
export const MakeKey = memo(({keyData})=>{
  return <>
    {keyData.map((v,i)=><div key={i} className={"key "+v[0]} data-seq={i}>
      {/* 키 윗면 */}
      <span className="span1 key-part">
      </span>
      {/* left */}
      <span className="span2"></span>
      {/* right */}
      <span className="span3"></span>
      {/* top */}
      <span className="span4"></span>
      {/* bottom */}
      <span className="span5"></span>
      {/* 키 맨윗면 */}
      <span className="span6"></span>
    </div>
  )}
</>;
}); /////////////// MakeKey ///////////////////
```

CSS

/* 키 단면 공통 */

```
.key span {
  position: absolute;
  width: 100%;
  height: 100%;
}
```

/* 앞면 */

```
.key span:nth-child(1) {
  background-color: #ccc;
  opacity: 0.9;
  width: calc(100% - 7.25px * 2);
  height: calc(56px - 7.25px * 2);
  top: 7.25px;
  left: 7.25px;
  border-radius: 1px;
  /* z축 밀기 */
  transform: translateZ(35.6px);
}
```

/* 오른쪽면 */

```
.key span:nth-child(2) {
  background-color: #ccc;
  opacity: 0.5;
  width: 37px;
  right: calc(37px / 2 * -1);
  /* y축 회전 */
  transform-origin: top;
  transform: rotateY(78.3deg) translateX(-18.4px);
  clip-path: polygon(100% 0%, 100% 100%, 0% calc(100% - 7.25px), 0% 7.25px);
}
```

하단 키보드 모델링

- 1) Position : absolute를 적용하고 top/left를 이동시켜 육면체 구성
- 2) 키보드의 형태를 이루기 위해 상단과 하단에 크기 차이 적용(7.25px)
- 3) 해당 길이만큼 clip-path로 측면 조정(사다리꼴 형태)
- 4) 조정 후 translateX와 rotateY로 각 면을 연결
- 5) Map을 이용해서 전체 키보드를 구성

페이지 구현 로그인 페이지-찾기 페이지, 회원가입 페이지

로그인페이지

1

LEOPOLD
Log-In

ID
Please enter your ID

Password
Please enter your Password

Did you forget id/password?
Don't have an account?

LOGIN

아이디 비밀번호 찾기

2

Find

< ID
Password >

Enter your infomation and we'll find your id.

User Name
Please enter your Name

Email
Please enter your Email

Don't have an account?

Find >

회원가입

3

LEOPOLD
Join Us

ID
Please enter your ID

Password
Please enter your Password

Confirm Password
Please enter your Confirm Password

User Name
Please enter your Name

Email
Please enter your Email

Are you already a member? Log In

SUBMIT

각 페이지는 하단의
안내 문구를 링크로
서로 이동 가능

로그인페이지

로컬스토리지에 회원정보를 저장하여 검색

- XSS를 막기위해 input에 정규식 검증
- 로컬스토리지 정보와 일치하면 로그인 후 로그인정보를 세션 스토리지에 저장

아이디 비밀번호 찾기

로컬스토리지에 회원정보를 저장하여 검색

- XSS를 막기위해 input에 정규식 검증
- 입력데이터와 회원정보를 검색하여 id는 보여주고 비밀번호는 초기화

회원가입

- XSS를 막기위해 input에 정규식 검증
- 입력데이터의 검증이 완료된 후 로컬스토리지에 저장

느낀점

1. React를 활용 시 re-render의 문제가 무수히 발생하였다.

이를 해결하기 위해서 랜더링 시점과 어떤 컴포넌트가 랜더링 되는 지 명확하게 알 필요가 있었다.

2. 스크롤 이벤트에서 위치 값 전달 시 값을 읽어오는 데 오류가 발생하여 null값이 전달되어 에러가 발생하였다.

에러가 발생하는 것을 막기 위해 이벤트를 비동기로 처리하고자 Promise()메서드 사용했다.

코드 가독성을 위해 aysnc, awite 문법도 활용해야 겠다.

3. 소셜 로그인, 도로명 주소 조회 등 오픈 API를 적극적으로 활용해야 겠다.