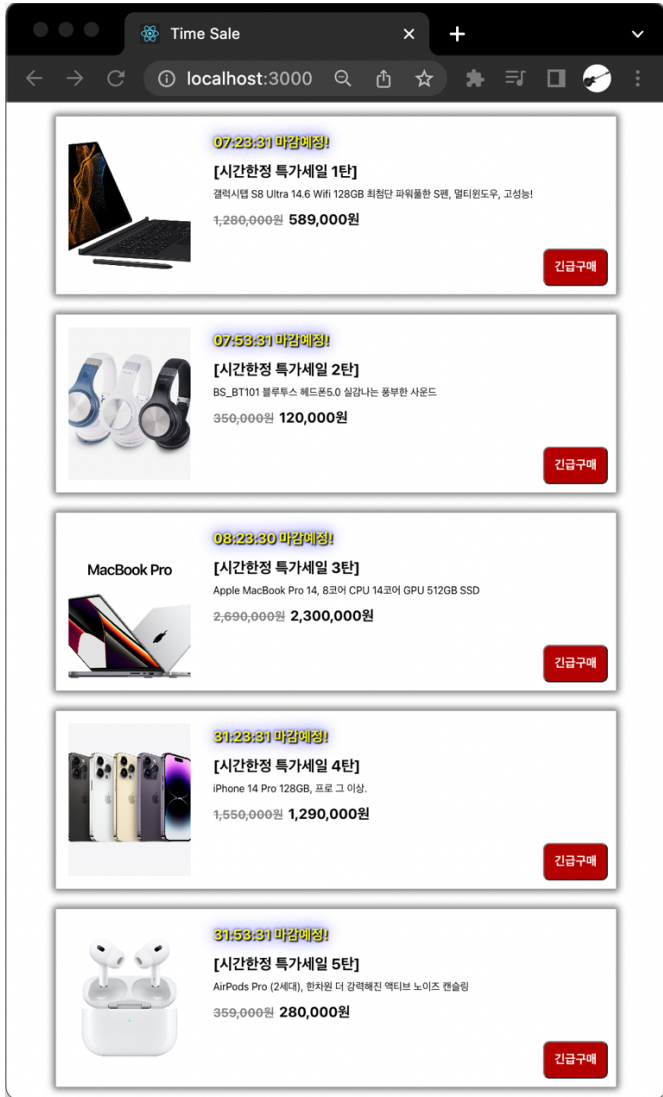
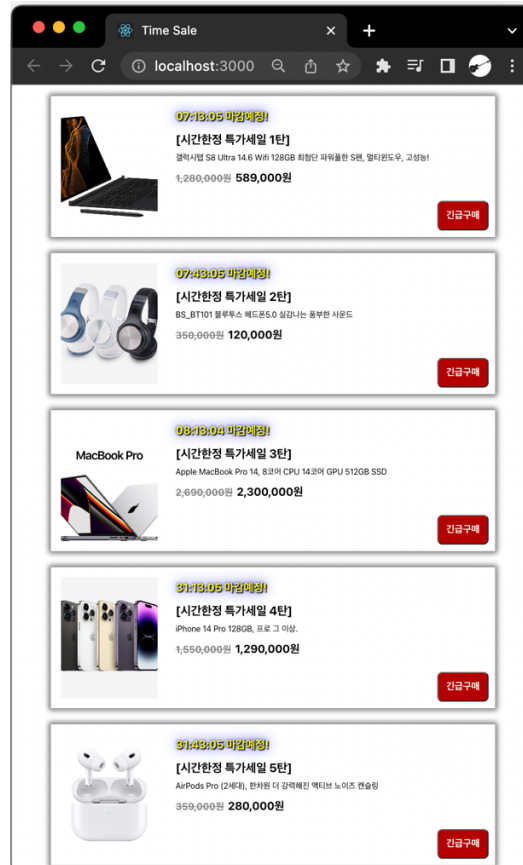


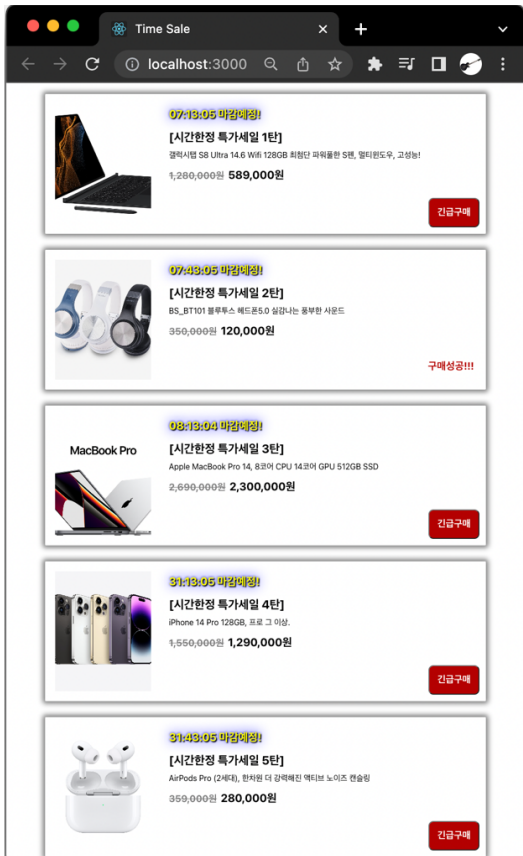
모바일 웹 실습일지 4

| 학번 | 201835745 | 이름 | 지서경 |
|------------------|--|----|-----|
| 주 의 사 항 | <p>- VS code 에서 create-react-app 을 이용하되 지정된 프로젝트명으로 작성할 것 - 반드시 react 코드를 사용할 것.</p> <p>- 결과가 맞다면, 브라우저화면을 캡처하고, 코드는 텍스트로 입력하여야.</p> <p>- 코드 텍스트의 포맷은 "맑은고딕 9pt"와 "줄간격 110%"으로 통일시킬 것</p> <p>- html 코드, js 코드, jsx 코드, css 코드에서 (줄단위로) 주석을 추가하여 코드를 설명하여야.</p> | | |
| 실 습 결 과 | <p>실습 1) 강의록 4-1 의 예제 2 에서 아래 조건을 반영/추가/수정하여 예제 2 프로그램을 보완하여야.</p> <p>조건 1) 예제 2 에서는 2 개 제품만 보였지만, 실습 1 에서는 총 5 개의 제품을 보이도록 수정하여야.</p> <p>조건 2) 수정할 때 *.map 함수를 사용한다.</p> <p>조건 3) 사용할 이미지, 사용할 제품내용등은 각각 달라야 하며, 인터넷쇼핑몰의 이미지와 내용을 참고하여 실제 제품과 같은 느낌이 들도록 수정하여야.</p> <p>조건 4) 5 개의 제품정보는 (강의록 3-2 의 예 2 와 같이) 별도의 파일(*.js)에서 읽어 처리할 수 있어야 한다.</p> <p>조건 5) 각 카드마다 맨 아래쪽 부분에 "긴급구매" 라는 버튼을 만든다.</p> <p>조건 6) 버튼을 누르면, "구매성공!!"라는 메시지가 (버튼이 없어지고) 버튼자리에 출력된다. (즉, 강의록 3-1 의 p.6 을 참조하여야)</p> | | |
| | <p><< 실행화면 캡처 (실행즉시 캡처) >></p>  | | |

<< 실행화면 캡처 (실행후 수초 지난후 캡처) >>>



< 실행화면 캡처 (두번째 제품에서 "구매성공"버튼을 누른 후 화면캡처) >>>



실
습
결
과

<< **index.html** 주식포함, 코드넣을 공간이 부족하면 양식을 더 만들어서 추가! >>

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <meta charset="utf-8" />
```

```
    <title>Time Sale</title>
```

```
  </head>
```

```
  <body>
```

```
    <div id="root"></div> <!--사용자 지정 컴포넌트들이 이곳에 렌더링됨-->
```

```
  </body>
```

```
</html>
```

<< 리액트 코드 (파일명: index.js) 주석포함, 코드넣을 공간이 부족하면 양식을 더 만들어서 추가! >>

```
import React from 'react'; //React 와 ReactDOM 임포트
```

```
import ReactDOM from 'react-dom/client';
```

```
import './index.css'; //index.js 에서 쓰일 css 스타일 코드 임포트
```

```
import TimeSale from './timesale' //timesale.jsx 에서 만든 Login 컴포넌트 임포트
```

```
const root = ReactDOM.createRoot(document.getElementById('root')); //index.html 의 div 태그를 id(root)로 가져옴
```

```
root.render( //root 에 렌더링 될 컴포넌트들 작성
```

```
  <React.StrictMode>
```

```
    <TimeSale /> {/* 임포트한 Login 컴포넌트 배치 (사용자 정의 태그) */}
```

```
  </React.StrictMode>
```

```
);
```

<< css 코드 (파일명: index.css) 주석포함, 코드넣을 공간이 부족하면 양식을 더 만들어서 추가! >>

```
body {  
  margin: 0; /* body 의 상하좌우측을 모두 브라우저에 붙임 (margin 없애기) */  
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',  
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',  
    sans-serif; /* 폰트 설정 */  
  /* 텍스트 브라우저에 부드럽게 렌더링 (webkit 기반의 브라우저에만 적용) */  
  -webkit-font-smoothing: antialiased;  
  -moz-osx-font-smoothing: grayscale;  
}
```

<< 리액트 코드 (파일명: product.js) 주석포함, 코드넣을 공간이 부족하면 양식을 더 만들어서 추가! >>

//각 카드 안에 디스플레이 될 상품 정보들

```
export const Products = [
```

```
  // 1. 갤럭시 탭 S8
```

```
  {
```

```
    img: "myfile.jpeg", //상품 이미지 경로
```

```
    time: "2022-9-21 23:00:00", //상품 판매가 종료되는 시간 (YYYY-MM-DD HH:mm:ss)
```

```
    title: "[시간한정 특가세일 1 탄]", //카드 제목
```

```
    description: "갤럭시탭 S8 Ultra 14.6 Wifi 128GB 최첨단 파워풀한 S 펜, 멀티윈도우, 고성능!", //상품 설명
```

```
    oldprice: "1,280,000 원", //세일 이전 가격
```

```
    price: "589,000 원", //세일 이후 가격 (판매가격)
```

```
  },
```

```
  // 2. BS_BT01 헤드폰
```

```
  {
```

```
    img: "myfile2.jpeg",
```

```
    time: "2022-9-21 23:30:00",
```

```
    title: "[시간한정 특가세일 2 탄]",
```

```
    description: "BS_BT101 블루투스 헤드폰 5.0 실감나는 풍부한 사운드",
```

```
    oldprice: "350,000 원",
```

```
    price: "120,000 원",
```

```
  },
```

```
  // 3. 맥북 프로 14
```

```
  {
```

```
    img: "myfile3.jpeg",
```

```
    time: "2022-9-21 23:59:59",
```

```
    title: "[시간한정 특가세일 3 탄]",
```

```
    description: "Apple MacBook Pro 14, 8 코어 CPU 14 코어 GPU 512GB SSD",
```

```
    oldprice: "2,690,000 원",
```

```
    price: "2,300,000 원",
```

```
  },
```

```
  // 4. 아이폰 프로 14
```

```
  {
```

```
    img: "myfile4.jpeg",
```

```
    time: "2022-9-22 23:00:00",
```

```
    title: "[시간한정 특가세일 4 탄]",
```

```
    description: "iPhone 14 Pro 128GB, 프로 그 이상.",
```

```
    oldprice: "1,550,000 원",
```

```
    price: "1,290,000 원",
```

```
  },
```

// 5. 에어팟 프로 2 세대

{

img: "myfile5.jpeg",

time: "2022-9-22 23:30:00",

title: "[시간한정 특가세일 5 탄]",

description: "AirPods Pro (2 세대), 한차원 더 강력해진 액티브 노이즈 캔슬링",

oldprice: "359,000 원",

price: "280,000 원",

},

]

<< 리액트 코드 (파일명: timesale.jsx) 주석포함, 코드넣을 공간이 부족하면 양식을 더 만들어서 추가! >>

```
import React from "react"; //React 임포트
import { useState } from "react"; //state 를 관리하기 위한 useState 함수 임포트
import { Products } from "../product"; //상품 정보들을 담고있는 Products 임포트
import "../timesale.css"; //timesale.jsx 에서 쓰일 css 스타일 코드 임포트

//마감시간을 인자로 받아와 현재시간에서 마감시간까지 남은 시간을 반환하는 함수
function getDeadline(endDate) {
  //safari 브라우저에서는 YYYY-MM-DD HH-MM-SS 형식을 지원하지 않음 (YYYY-MM-DDTHH:mm:ss.sssZ 형식
  //사용)
  //safari 브라우저에서도 시간이 잘 보일 수 있도록 정규식을 사용해 포맷 변경 필요
  endDate = endDate.replace(/-/g, ""); //포맷 변경

  let dead = new Date(endDate); //마감시간 설정
  let current = new Date(); //현재시간 설정
  let deadline = dead.getTime() - current.getTime(); //현재시간에서 마감까지 남은 시간 계산
  let hour = Math.floor(deadline / (1000*60*60)); //시
  let min = Math.floor((deadline % (1000*60*60)) / (1000*60)); //분
  let sec = Math.floor(((deadline % (1000*60*60)) % (1000*60)) / 1000); //초

  //한 자리 수의 시간일 때 앞에 0 을 붙여 두 자리 수로 맞춰주기 위해 ('0' + 시간.toString(10)).slice(-2)를 해줌
  return ('0'+hour.toString(10)).slice(-2) + ":" + ('0'+min.toString(10)).slice(-2) + ":" +
  ('0'+sec.toString(10)).slice(-2); //10 진수 문자열로 표기
}

//상품 정보를 담고있는 Card 컴포넌트
function Card(props) {
  //useState 를 이용하여 구매 버튼 클릭여부(state) 기본값 파라미터로 넣어서 호출
  const [purchased, setPurchased] = useState(false); //purchased 는 현재 상태(false), setPurchased 는
  purchased 를 최신 상태로 설정해주는 setter 함수

  //구매 버튼 클릭 시 호출되는 함수
  function buyProduct() {
    setPurchased(true); //purchased 의 state 가 false 에서 true 로 바뀜
  }

  return(
    //props 를 통해 TimeSale 컴포넌트에서 상품 정보를 받아옴
    <div className="card"> {/* div 태그 (Card 컴포넌트 전체를 감쌈) */}
      <img src={props.img} className="cardimg" alt="product" /> {/* img 태그 (상품 이미지) */}
      <div className="cardbody"> {/* div 태그 (상품의 상세정보를 담는 박스) */}
        <h2 className="cardtime">{props.time} 마감예정!</h2> {/* h2 태그 (마감시간) */}
        <h2 className="cardtitle">{props.title}</h2> {/* h2 태그 (카드 제목) */}
      </div>
    </div>
  );
}
```



```

        <p className="carddescription">{props.description}</p> {/* p 태그 (상품설명) */}
        <h3 className="cardprice"> {/* h3 태그 (상품가격) */}
            <del style={{color: "gray"}}>{props.oldprice}</del>{props.price} {/* del 태그 (상품의 이전가격에
줄이 그어져 있음), 상품의 세일가격 */}
        </h3>
    </div>
    <div className="btnwrapper"> {/* div 태그 (구매버튼을 담는 박스) */}
        {/* 렌더링되는 시점에 함수가 호출되지 않도록 on 이벤트이름={실행하고싶은함수} 형태로 이벤트
설정 */}
        {/* 삼항연산자(a?x:y)를 사용하여 purchased 의 값이 true 면 구매성공 텍스트가, false 면 긴급구매
버튼이 나타나도록 해줌 */}
        {purchased ? <p className="purchasetxt">구매성공!!!</p> : <button className="cardbutton"
onClick={buyProduct}>긴급구매</button>}
    </div>
</div>
);
}

//반복되는 Card 들을 하나씩 배치하는 컴포넌트
function TimeSale(props) {
    return(
        <div className="wrapper"> {/* div 태그 (컴포넌트 전체를 감쌈) */}
            {/* map 함수를 이용해 반복되는 컴포넌트 렌더링 */}
            {/* Products 에서 상품 정보를 받아와 map 함수를 이용해 index 를 key 로 item 을 불러오고 item 의 각
요소들을 배치함 */}
            {Products.map((item, index) => {
                return (
                    <Card
                        key={index} //index 를 key 로 사용해 어떤 원소가 변경되었는지 빠르게 감지하고 이에 맞는
item 을 불러옴
                        img={item.img} //Products 의 index 번째 item 의 img(item.img)를 img 에 넣어줌
                        time={getDeadline(item.time)} //Products 의 index 번째 item 의 time(item.time)을 time 에
넣어줌 (getDeadline 함수를 이용해 마감시간까지 남은 시간을 넣어줌)
                        title={item.title} //Products 의 index 번째 item 의 title(item.title)을 title 에 넣어줌
                        description={item.description} //Products 의 index 번째 item 의
description(item.description)을 description 에 넣어줌
                        oldprice={item.oldprice} //Products 의 index 번째 item 의 oldprice(item.oldprice)를 oldprice 에
넣어줌
                        price={item.price} //Products 의 index 번째 item 의 price(item.price)를 price 에 넣어줌
                    />
                )
            })}
        </div>
    );
}

```

```
);  
}
```

```
export default TimeSale; //TimeSale 컴포넌트 (element 객체) 내보냄
```

<< css 코드 (파일명: timesale.css), 코드넣을 공간이 부족하면 양식을 더 만들어서 추가! >>

```
.wrapper {
  margin: 2rem; /* 카드 여백 크기 */
  display: grid; /* 각 그리드 아이템들을 그리드 형태로 배치 */
  gap: 2rem; /* 그리드 아이템들 사이의 여백 설정 */
  grid-template-rows: 1fr; /* 그리드 아이템을 동일 크기 행으로 배치 */
  justify-content: center; /* 그리드 아이템들 가로 중앙 정렬 */
}

.card {
  box-shadow: 0 0 10px 5px gray; /* 3 차원 박스 효과 */
  border-radius: 2px; /* 카드 모서리 약간 둥글게 설정 */
  min-width: 18rem; /* 카드 최소 크기 */
  display: flex; /* 아이템들 flex 형태로 배치 */
}

.carding {
  width: 12rem; /* 상품이미지의 가로폭, 세로폭 설정 */
  height: 15rem;
  object-fit: cover; /* 이미지를 가로세로에 맞게 채움 */
  padding: 20px; /* 이미지 박스 안의 여백 설정 */
}

.cardbody {
  margin: 1rem; /* 상품 상세정보 박스 밖의 여백 설정 */
}

.cardtime {
  font-size: 1.4rem; /* 폰트 크기 설정 */
  color: yellow; /* 폰트 색상 설정 */
  margin-top: 0.6rem; /* 바깥 위쪽 여백 설정 (border 와 바깥과의 여백) */
  text-shadow: 2px 2px 2px black, 0 0 25px blue, 0 0 5px darkblue; /* 글씨 테두리에 그림자 효과 설정 */
}

.cardtitle {
  line-height: 1.4rem; /* 텍스트 줄간격 설정 */
  margin-bottom: 1rem; /* 바깥 아래쪽 여백 설정 */
}

.caddescription {
  line-height: 1.2rem; /* 텍스트 줄간격 설정 */
}
```

```
.cardprice {
  font-size: 1.4rem; /* 폰트 크기 설정 */
  margin-top: 1.5rem; /* 바깥 위쪽 여백 설정 */
}

.cardprice del {
  font-size: 1.2rem; /* 폰트 크기 설정 */
  margin-right: 0.5rem; /* 바깥 오른쪽 여백 설정 */
}

.btnwrapper {
  margin: auto 0.8rem 0.8rem auto; /* 버튼이 맨 오른쪽 아래에 위치할 수 있도록 margin 설정 (상, 우, 하, 좌) */
}

.purchasetxt {
  font-size: 1.3rem; /* 폰트 크기 설정 */
  font-weight: bold; /* 글씨 굵게 설정 */
  color: rgb(180, 0, 0); /* 폰트 색상 설정 */
  padding-right: 0.7rem; /* 내부 오른쪽 여백 설정 (본문과 border 사이의 여백) */
}

.cardbutton {
  font-size: 1.2rem; /* 폰트 크기 설정 */
  font-weight: bold; /* 글씨 굵게 설정 */
  color: white; /* 폰트 색상 설정 */
  border-radius: 10px; /* 버튼 모서리 둥글게 설정 */
  background-color: rgb(180, 0, 0); /* 버튼 색상 설정 */
  padding: 1rem; /* 버튼 안쪽 여백 설정 */
}
```