react로 경매 카운터 기능 구현

날짜	@2024년 7월 27일
진행 프로젝트	FreshTrash

[1단계] State 초기화 (`useState`)

- 컴포넌트는 `timeLeft`라는 상태를 가지고 있으며, 이는 남은 시간을 저장합니다.
- 컴포넌트가 처음 렌더링될 때, `calculateTimeLeft` 함수를 호출하여 초기 `timeLeft` 값을 설정합니다.
- startDate와 endDate는 자신이 만드는 웹페이지 데이터에 맞게 `calculateTimeLeft` 함수의 인자로 받아오면 됩니다.

```
const [timeLeft, setTimeLeft] = useState(
  calculateTimeLeft(startDate, endDate),
);
```

[2단계] 타이머 설정 (`useEffect`)

- `useEffect` 훅을 사용하여 컴포넌트가 마운트될 때 타이머를 설정하고, 매초마다 `timeLeft`를 업데이트합니다.
- 컴포넌트가 언마운트될 때 타이머를 정리합니다.

```
useEffect(() => {
//1초마다 남은 시간을 업데이트하는 타이머 설정
   const timer = setInterval(() => {
     setTimeLeft(calculateTimeLeft(startDate, endDate));
   }, 1000);

   //컴포넌트가 언마운트 될때 타이머 정리
   return () => clearInterval(timer);
 }, [startDate, endDate]);
```

▼ `setInterval` 함수

일정 시간 간격을 두고 함수를 실행하는 함수이다.

따라서 여기서는 매 1초(1000밀리초)마다 `calculateTimeLeft` 함수를 호출하여 남은 시간을 계산하고, 이를 `setTimeLeft` 에 상태를 업데이트 한다.

즉, 아래와 같다.

일정 시간 간격 ⇒ 1초

반복 → `calculateTimeLeft` 함수를 호출하여 남은 시간을 계산하고, 이를 `setTimeLeft` 에 상태를 업데이트

▼ `clearInterval` 함수

컴포넌트가 언마운트 될때, 즉 컴폰너트가 DOM에서 제거될때 타이머나 다른 비동기 작어을 중지하야 메모리 누수를 막을 수 있기 때문에 useEffect 훅을 통해 클린업 함수를 사용하여 작성한다.

여기서 사용된 클린업 함수가 clearInterval함수이고 이 함수는

이전에 `setInterval()`을 호출하여 생성한 타이머에 의해 반복되는 작업을 취소다.

[3단계] 시간 계산 함수 (`calculateTimeLeft`)

> 💡 시간 계산 함수는 사용하고자 하는 목적에 맞게 구현하시면 됩니다.
> 이 글은 경매 기준으로 작성되었습니다.

- 날짜 객체 생성

```
const start = new Date(startDate); //경매 시작 날짜
const end = new Date(endDate); //경매 종료 날짜
const now = new Date(); //현재 날짜
```

- 남은 시간을 저장할 객체 초기화

```
let timeLeft = {};
```

- 경매 시작 날짜와 현재 날짜 비교

```
//아직 경매 시작 날짜가 되지 않았다면 'OPEN 시작날짜'화면에 띄운다.
if (now < start) {
  timeLeft = {
    message: `OPEN ${startDate}`,
  };
}
```

- 경매 종료 날짜와 현재 날짜 비교

▼ 현재 날짜가가 종료날짜보다 이전인 경우

```
else if (now < end) {
      const difference = end - now;
      //남은 일, 시간, 분, 초를 계산하여 timeLeft 객체에 저장한다.
      timeLeft = {
        days: Math.floor(difference / (1000 * 60 * 60 * 24)),
        hours: Math.floor((difference / (1000 * 60 * 60)) % 24),
        minutes: Math.floor((difference / 1000 / 60) % 60),
        seconds: Math.floor((difference / 1000) % 60),
      };
    }
```

▼ 로직 자세히 설명

이 로직이 필요한 이유?

`difference` 값만 출력하는 경우

```
const difference = end - now;
console.log(difference);
```

이렇게 하면, 예를 들어 3일 2시간 5분 10초가 남았을 때, 출력 값은 다음과 같습니다:

```
265510000 // 밀리초
```

로직 설명

남은 일 계산

```
days: Math.floor(difference / (1000 * 60 * 60 * 24)),
```

- `difference`를 하루의 밀리초 수로 나눕니다.
  - 하루는 `1000` 밀리초(1초) × `60` 초(1분) × `60` 분(1시간) × `24` 시간 = `86400000` 밀리초.

- `difference / 86400000`은 남은 일 수를 소수점까지 계산합니다.
- `Math.floor`를 사용하여 소수점을 버리고 정수 부분만 남깁니다.

남은 시간 계산

```
hours: Math.floor((difference / (1000 * 60 * 60)) % 24),
```

- `difference`를 한 시간의 밀리초 수로 나눕니다.
  - 한 시간은 `1000` 밀리초 × `60` 초 × `60` 분 = `3600000` 밀리초.
- `difference / 3600000`은 남은 시간을 소수점까지 계산합니다.
- 이를 `24`로 나눈 나머지(`% 24`)를 구합니다. 이는 24시간을 초과하는 부분을 제거하고 남은 시간만 계산합니다.
- `Math.floor`를 사용하여 소수점을 버리고 정수 부분만 남깁니다.

남은 분 계산

```
minutes: Math.floor((difference / 1000 / 60) % 60),
```

- `difference`를 1분의 밀리초 수로 나눕니다.
  - 1분은 `1000` 밀리초 × `60` 초 = `60000` 밀리초.
- `difference / 60000`은 남은 분을 소수점까지 계산합니다.
- 이를 `60`으로 나눈 나머지(`% 60`)를 구합니다. 이는 60분을 초과하는 부분을 제거하고 남은 분만 계산합니다.
- `Math.floor`를 사용하여 소수점을 버리고 정수 부분만 남깁니다.

남은 초 계산

```
seconds: Math.floor((difference / 1000) % 60),
```

- `difference`를 1초의 밀리초 수로 나눕니다.
  - 1초는 `1000` 밀리초.
- `difference / 1000`은 남은 초를 소수점까지 계산합니다.
- 이를 `60`으로 나눈 나머지(`% 60`)를 구합니다. 이는 60초를 초과하는 부분을 제거하고 남은 초만 계산합니다.
- `Math.floor`를 사용하여 소수점을 버리고 정수 부분만 남깁니다.

▼ 종료날짜가 지난 경우

```
//종료날짜가 지난 경우 'CLOSE'라는 문구를 화면에 띄운다.
else {
      timeLeft = {
        message: 'CLOSE',
      };
    }
```

- timeLeft 반환
  - 남은 시간이 `message` 속성에 저장된 경우, 그 메시지를 표시합니다.
  - 그렇지 않은 경우, 남은 일, 시간, 분, 초를 각각 표시합니다.

```
  return timeLeft;
```

- `calculateTimeLeft` 함수 전체코드

```
function calculateTimeLeft(startDate, endDate) {
   const start = new Date(startDate);
   const end = new Date(endDate);
   const now = new Date();

   let timeLeft = {};
```

```
  if (now < start) {
    timeLeft = {
      message: `OPEN ${startDate}`,
    };
  } else if (now < end) {
    const difference = end - now;
    timeLeft = {
      days: Math.floor(difference / (1000 * 60 * 60 * 24)),
      hours: Math.floor((difference / (1000 * 60 * 60)) % 24),
      minutes: Math.floor((difference / 1000 / 60) % 60),
      seconds: Math.floor((difference / 1000) % 60),
    };
  } else {
    timeLeft = {
      message: 'CLOSE',
    };
  }

  return timeLeft;
}
```

전체코드

```
import React, { useEffect, useState } from 'react';

const DateCounter = ({ startDate, endDate }) => {
  const [timeLeft, setTimeLeft] = useState(
    calculateTimeLeft(startDate, endDate),
  );

  useEffect(() => {
    const timer = setInterval(() => {
      setTimeLeft(calculateTimeLeft(startDate, endDate));
    }, 1000);
    return () => clearInterval(timer);
  }, [startDate, endDate]);

  function calculateTimeLeft(startDate, endDate) {
    const start = new Date(startDate);
    const end = new Date(endDate);
    const now = new Date();

    let timeLeft = {};

    if (now < start) {
      timeLeft = {
        message: `OPEN ${startDate}`,
      };
    } else if (now < end) {
      const difference = end - now;
      timeLeft = {
        days: Math.floor(difference / (1000 * 60 * 60 * 24)),
        hours: Math.floor((difference / (1000 * 60 * 60)) % 24),
        minutes: Math.floor((difference / 1000 / 60) % 60),
        seconds: Math.floor((difference / 1000) % 60),
      };
```

```
    } else {
      timeLeft = {
        message: 'CLOSE',
      };
    }

    return timeLeft;
  }

  return (
    <div>
      {timeLeft.message ? (
        <div className="text-gray-400 font-semibold">{timeLeft.message}</div>
      ) : (
        <div className="text-gray-400 font-semibold">
          {timeLeft.days}일 {timeLeft.hours}시간 {timeLeft.minutes}분{' '}
          {timeLeft.seconds}초
        </div>
      )}
    </div>
  );
};

export default DateCounter;
```