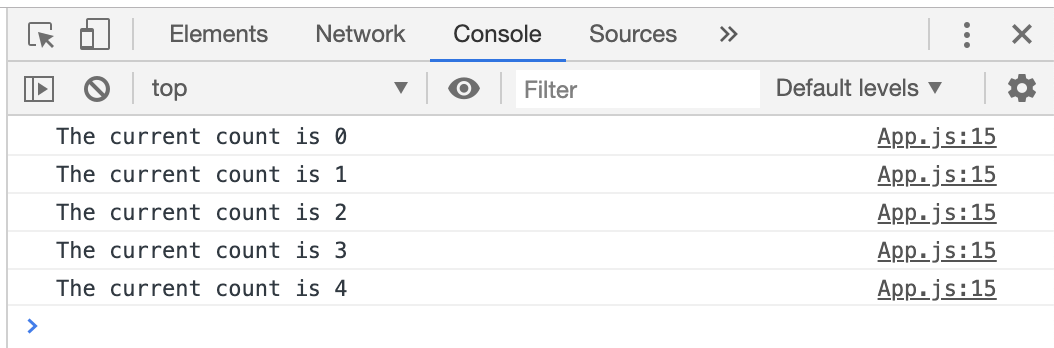
1. 리액트 디버깅 스킬을 아는대로 쓰시오.

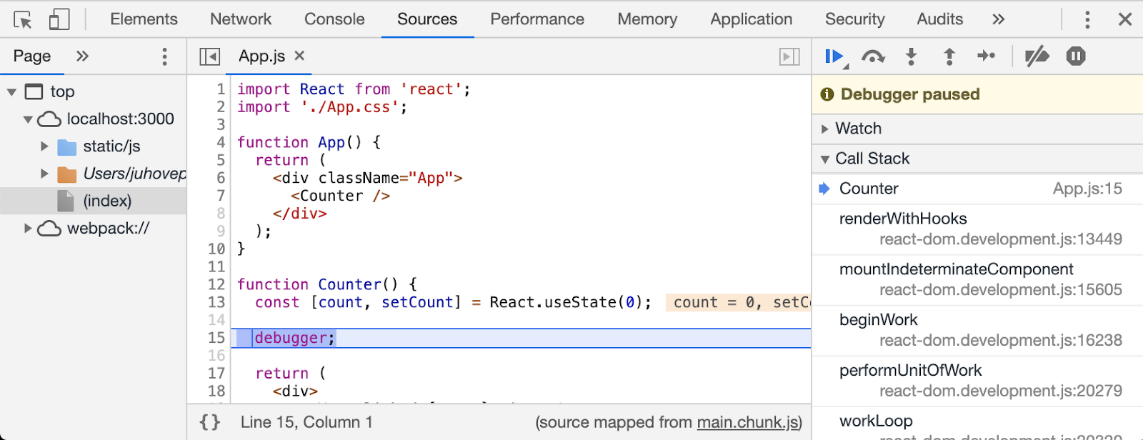
1. console 도구 활용한 log 찍어보기

* 빠르고 로직을 이해하기 쉽다



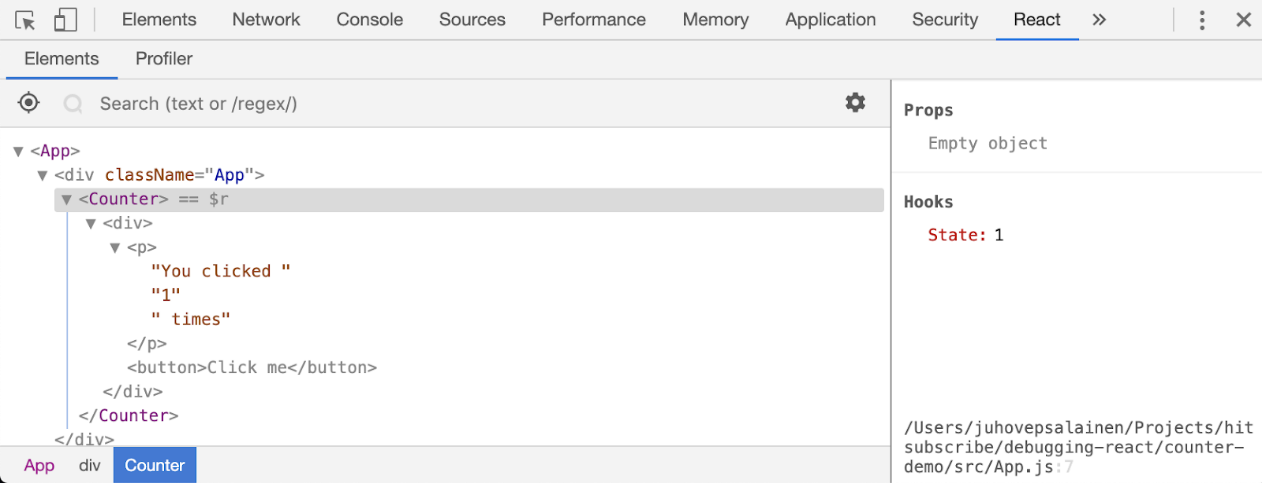
1. 브라우저 개발자도구 - debugger 활용

* 실행 구문에 debugger; 넣으면 해당 위치에서 실행됨



* 코드에 breakpoint를 지정함으로써 코드 실행 과정을 쪼개서 분석 가능
* 이미 실행한 구문의 결과값 확인
* Call Stack을 통해서 디버깅 위치까지 코드가 실행된 과정 확인

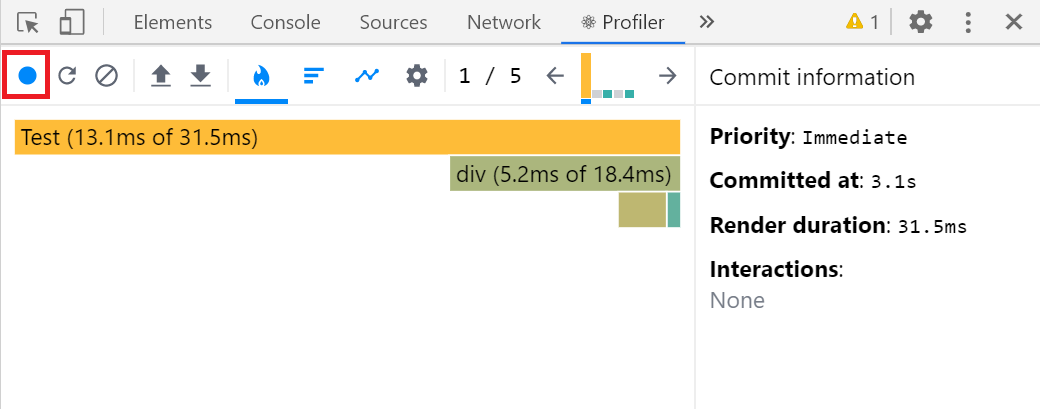
1. React DevTool 활용
2. component 탭



* + 애플리케이션의 Element 구조와 해당 요소의 state 값 등 확인 가능

1. profiler 탭

* 리액트 애플리케이션 퍼포먼스/ 기능 구현 관련 문제 확인 가능



1. 기록 버튼 클릭
2. 애플리케이션 실행
3. 기록 버튼 클릭
4. 네트워크 관련 문제 디버깅

* 개발자 도구 Networking 탭 이용
* 코드의 적절한 위치에서 debugger 기능 실행하면 시너지 효과

1. Redux 디버깅: Redux DevTools Extension 이용
2. React regression test

* 이전에는 잘 작동했으나 어느 시점 이후부터 오류가 발생했을 때 어디서 오류가 발생했는지 확인 가능
* Git으로 버전 관리해온 경우에 git bisect 커맨드 활용 가능

1. git bisect good [버전] 입력하여 문제 없었던 버전 알려주기
2. git bisect start 입력하여 프로세스 시작 → git이 지금 버전과 문제 없었던 버전의 중간 버전 실행하고 잘 작동하는 지 확인 → git biset good 또는 bad로 결과 입력
3. 기능이 망가진 부분 찾아낼 수 있게 됨 → 해당 수정이 포함된 commit 내용 분석하여 디버깅

* git commit 하는 과정에서 npm 버전이 바뀌었다면 commit 했을 당시 버전과 git bisect 실행하는 npm 버전 맞춰줘야 함

2. document load event와 DOMContentLoaded event의 차이점은 무엇인가요?

DOMContentLoaded 이벤트

DOMContentLoaded 이벤트는 초기 HTML 도큐먼트가 로드되고 파스되면

스타일시트, 이미지 및 다른 프레임이 로딩되기전에 실행됩니다. 이벤트의 진짜

타겟은 로드가 완료된 이벤트 입니다. 이벤트는 Window 인터페이스에서 이벤트를

수신하여 캡쳐하고 버블링할 수 있습니다. 자세한 내용은 Document:

DOMContentLoaded 참고부탁드립니다.

동기식 javascript는 DOM을 파서하는 것을 정지합니다. 만약 유저가 페이지를

요청했을 때 DOM이 최대한 빠르게 파서되기를 원한다면 javascript를

비동기식으로 변환하든지 스타일시트를 수정하면 됩니다. 보통 스타일시트는 병렬로

로드되면서 DOM 파싱속도를 저하하고 주요 HTML 도큐먼트 트래픽을 스틸합니다.

load 이벤트

load 이벤트는 스타일시트 이미지와 같이 의존성이 있는 리소스가 준비되면

실행됩니다. 이것은 DOMContentLoaded와 정반대입니다.

3. jsx문법에 대해 아는대로 쓰시오.

**01. 감싸인 요소**

컴포넌트에 여러 요소가 있다면 반드시 부모 요소로 감싸주어야 한다.

**\*왜 리액트 컴포넌트 에서 부모 요소로 여러 개의 요소를 감싸야 할까?**VirtualDOM 에서 컴포넌트 변화를 감지해 낼 때 효율적으로 비교할 수 있도록 하나의 DOM 트리 구조로 이루어져야 한다는 규칙이 있기 때문이다.

**리액트 v16 이상부터 도입된 Fragment 는 div 요소 대신 사용할 수 있다.**  
import 구문에서 react 모듈에 있는 Fragment 라는 컴포넌트를 추가로 불러온다.  
Fragment 는 <> 과 같은 형태로도 표현할 수 있다.

**02. 자바스크립트 표현**

JSX 안에서는 자바스크립트 표현식을 쓸 수 있다. JSX 내부에서 코드를 { } 로 감싸면 된다.

**03. if 문 대신 조건부 연산자**

JSX 내부의 자바스크립트에서는 if 문을 사용할 수 없다.

대신 JSX 밖에서 if 문을 사용에 값을 설정하거나, { } 안에 조건부 연산자(삼항연산자)를 사용하면 된다.

**04. AND 연산자(&&)를 사용한 조건부 렌더링**

&& 연산자 로 조건부 렌더링을 할 수 있는 이유는 리액트에서 false를 렌더링할 때는 null 과 마찬가지로 아무것도 나타나지 않기 때문이다.

**05. undefined 를 렌더링하지 않기**

리액트 컴포넌트에서는 함수에서 undefined 만 반환해 렌더링하는 상황을 만들면 안된다. (오류 발생) 만약 어떤 값이 undefined일 수도 있다면 OR (||) 연산자를 사용해 오류를 방지할 수 있다. 반면 JSX 내부에서는 undefined를 렌더링해도 괜찮다.

**06. 인라인 스타일링**

리액트에서는 DOM 요소에 스타일을 적용할 때는 문자열 형태가 아닌 객체 형태로 넣어줘야 한다.

스타일 이름 중에 background-color 처럼 - 문자가 포함되는 이름은 - 을 제거해 카멜 표기법으로 작성해야 한다.

즉, background-color 이 backgroudColor 로 작성된다.

**07. class 대신 className**

JSX 에서는 class 가 아닌 className 으로 설정해 주어야 한다. className 이 아닌 class 값을 설정해도 스타일이 적용되기는 하지만 console 에 경고가 나타난다.

(리액트 v16 이상부터는 class 를 className으로 변환시켜 주고 경고를 띄운다.)

**08. 꼭 닫아야 하는 태그**

JSX 에서는 태그를 닫지 않으면 오류가 발생한다.

**09. 주석**

JSX 안에서 주석을 작성하는 방법은 일반 자바스크립트에서 작성하는 방법과는 다르다.

import React from ‘react’;

function App() {

return (

{/\* JSX 안에서 주석은 이렇게 작성한다. \*/}

<div className=“react” // 이런식으로 주석을 작성할 수도 있다.

>{name}</div>

// 하지만 이런 주석이나

/\* 이런 주석은 페이지에 그대로 나타난다. \*/

);

}

export default App;