PRE-ONBOARDING

면접 단골 질문
이벤트 루프 다시 돌아보기

FRONTEND CHALLENGE 25차

wanted

면접 단골 질문
이벤트 루프 다시 돌아보기

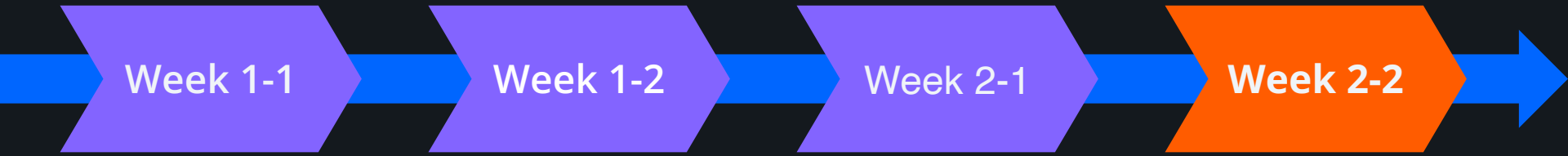
Week 2-2

면접 질문 발표해보기

himprover

wanted

강의 진행 미리보기



- 강의 OT
- 이벤트 루프 개념
- 비동기, 동기
- 싱글, 멀티 스레드

- 큐, 스택
- 이벤트루프 큐, 스택

- callback, promise
- async/await
- 이벤트 루프 고려한 코드

- 면접 질문 발표해보기

목차

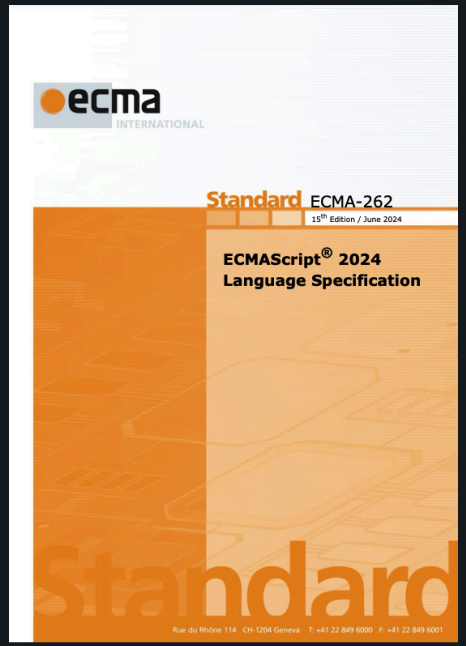
3시간 미리보기

1. ES5 에서 ES6 으로의 변화점
2. SPA, CSR, SSR
3. React 16+의 동작 원리
4. 비동기 처리에서 고려해야 할 점
5. Event Loop 정리
6. Engineer가 가져야 하는 습관

wanted

1. ES5에서 ES6로

ES5에서 ES6로 발표해보기



Speaker

9월 프론트엔드 챌린저 조민지님

ES5에서 ES6

- ECMAScript가 정확히 무엇인가
- ES5에서 ES6로 넘어 오며 변화된 요소의 이유는 무엇인가

wanted

2. SPA, CSR, SSR

SPA, CSR, SSR 발표해보기

Speaker

9월 프론트엔드 챌린저 이경택님

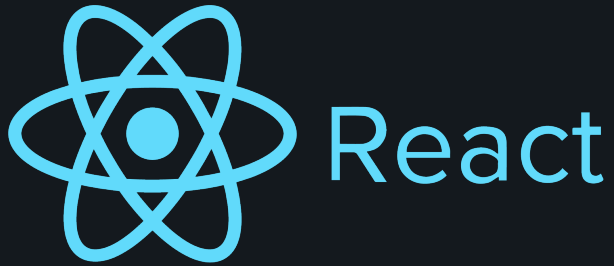
SPA, CSR, SSR

- 각각은 어떤 개념을 가지는가
- 기술 스택을 선택 할 때 어떤 점을 고려해야 하는가

wanted

3. React 16+ 동작원리

React 16+ 동작원리 발표해보기



Speaker

9월 프론트엔드 챌린저 심민섭님

React 16+ 의 Fiber Architecture

- **React 16 버전부터 변화된 가장 큰 점은 무엇인가**
- **그 변화는 무엇 때문에 일어났는가**
- **그것을 통해 어떤 이점을 얻을 수 있는가**

wanted

4. 비동기 처리에서 고려해야 할 점

JavaScript의 Promise.all

Promise.all( )

Promise.allSettled( )

API waterfall 방지

JavaScript의 Promise.all

유저 정보 API

현재 주차 정보 API　userId

주차 기록 정보 API　userId　parkingId

소개 페이지
(/service/parking/landing)

JavaScript의 Promise.all

유저 정보 API

현재 주차 정보 API

userId

주차 기록 정보 API

userId

parkingId

```
const userId = await getUserData();
const parkingId = await getActiveParking(userId);
const history = await getParkingHistory(userId, parkingId);

trackingEvent(userId, parkingId, history);
```

소개 페이지

(/service/parking/landing)

JavaScript의 Promise.all

1. 유저 정보 API
2. 현재 주차 정보 API — userId
3. 주차 기록 정보 API — userId, parkingId

```
const userId = await getUserData();
const parkingId = await getActiveParking(userId);
const history = await getParkingHistory(userId, parkingId);

trackingEvent(userId, parkingId, history);
```

소개 페이지
(/service/parking/landing)

JavaScript의 Promise.all

유저 정보 API

오늘 날씨 API

A 주차장 남은 대수 API

이벤트 남은기간 조회 API

소개 페이지
(/service/parking/landing)

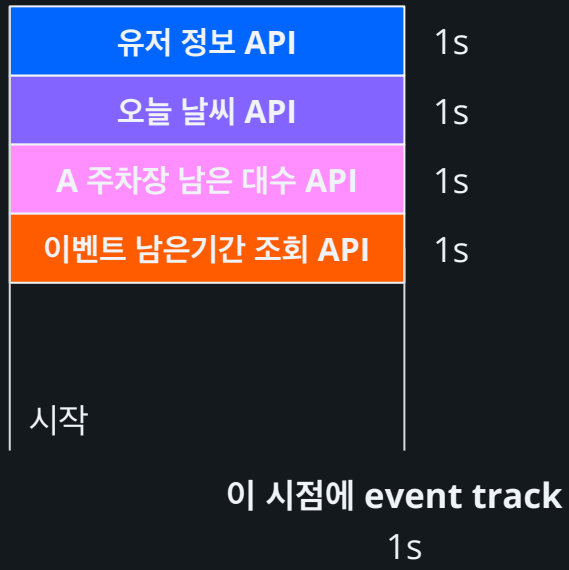
JavaScript의 Promise.all

```
const userId = await getUserData();
const weatherData = await getTodayWeather();
const leftSpace = await getAParkZoneLeftSpace();
const leftEvent = await getEventLeftDay();

trackingEvent(userId, weatherData, leftSpace, leftEvent);
```

(/service/parking/landing)

JavaScript의 Promise.all

JavaScript의 Promise.all



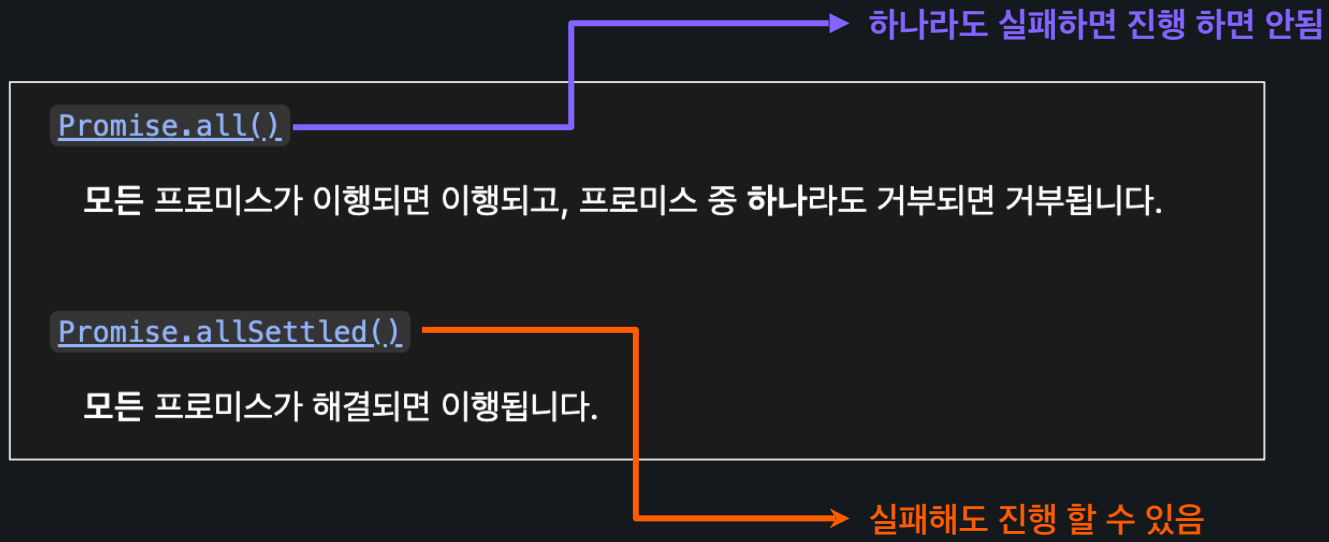
4s ▶ 1s

3s 절약됨

JavaScript의 Promise.all

```
const [userId, weatherData, leftSpace, leftEvent] = await Promise.all([getUserData(), getTodayWeather(),
                                                                        getAParkZoneLeftSpace(), getEventLeftDat()]);

trackingEvent(userId, weatherData, leftSpace, leftEvent);
```

JavaScript의 Promise.all



JavaScript의 Promise.all

```
const userId = await getUserData();
const weatherData = await getTodayWeather();
const leftSpace = await getAParkZoneLeftSpace();
const leftEvent = await getEventLeftDay();

trackingEvent(userId, weatherData, leftSpace, leftEvent);
```

```
const userId = getUserData();
const weatherData = getTodayWeather();
const leftSpace = getAParkZoneLeftSpace();
const leftEvent = getEventLeftDat();

trackingEvent(await userId, await weatherData, await leftSpace, await leftEvent);
```

JavaScript의 Promise.all

getUserData fetching 종료를 기다리고 다음 fetch 요청이 시작됨

```
const userId = await getUserData();
const weatherData = await getTodayWeather();
const leftSpace = await getAParkZoneLeftSpace();
const leftEvent = await getEventLeftDay();

trackingEvent(userId, weatherData, leftSpace, leftEvent);
```

일단 모두 fetch 요청을 보낸 후 trackingEvent 시점에서 기다림

```
const userId = getUserData();
const weatherData = getTodayWeather();
const leftSpace = getAParkZoneLeftSpace();
const leftEvent = getEventLeftDat();

trackingEvent(await userId, await weatherData, await leftSpace, await leftEvent);
```

wanted

5. Event Loop 정리

Event Loop란 무엇인가

이벤트 루프란 **싱글 스레드**로 동작하는 **JavaScript**를 **브라우저**에서 **동시성**을 제공하기 위한 **동작 방식을 의미**

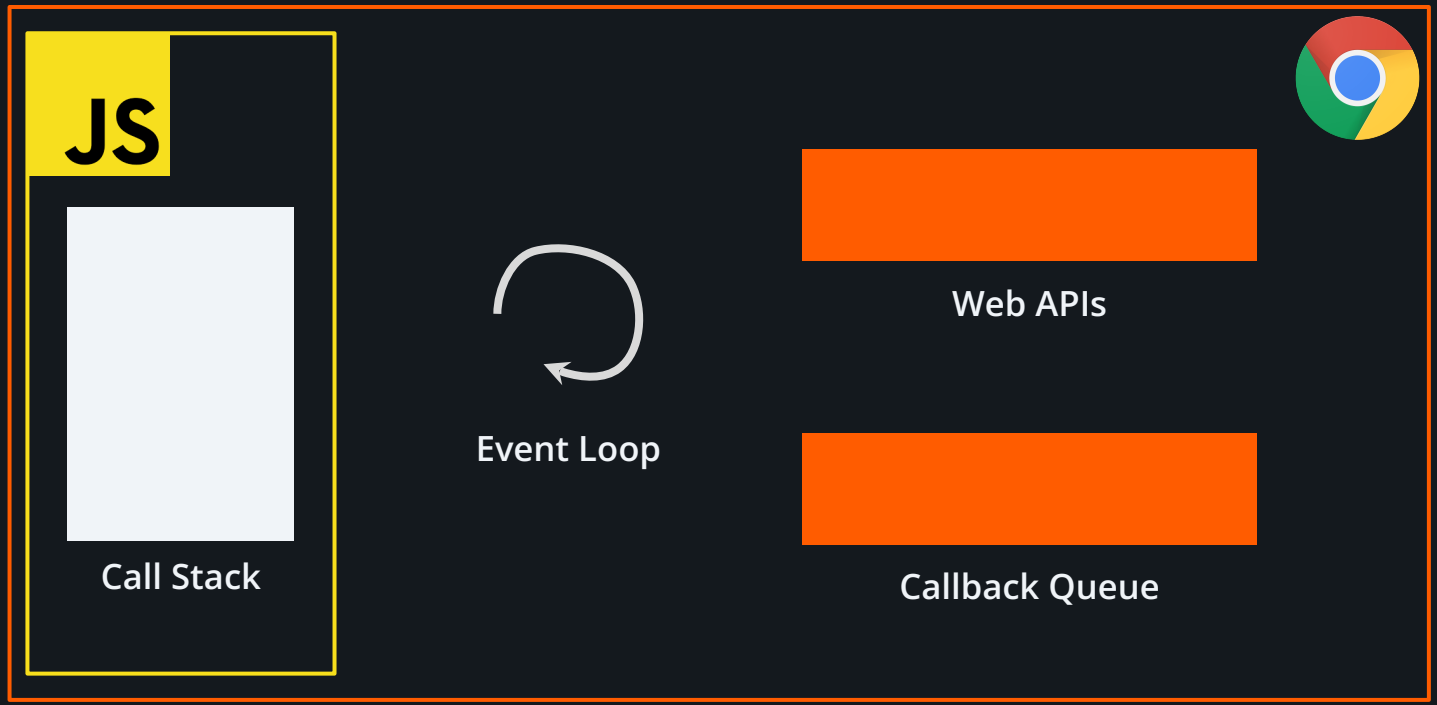
wanted

Event Loop란 무엇인가

JavaScript는 싱글스레드 이다

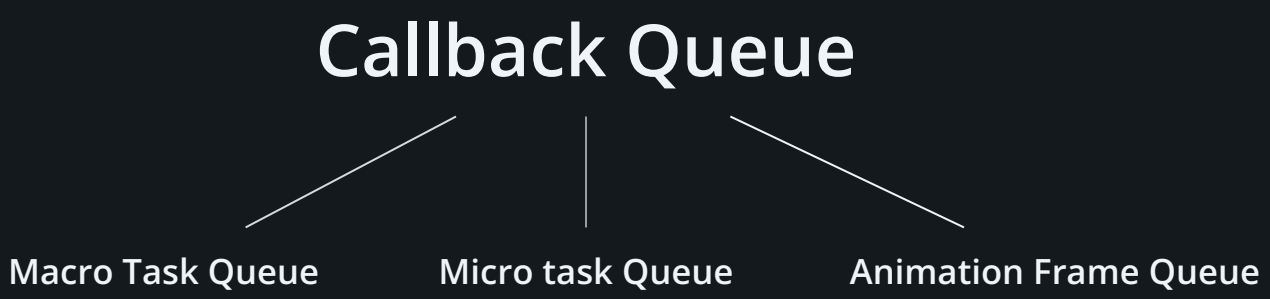
JavaScript Runtime은 싱글스레드 이다

Event Loop란 무엇인가



면접 단골 질문 이벤트 루프 다시 돌아보기

Event Loop란 무엇인가

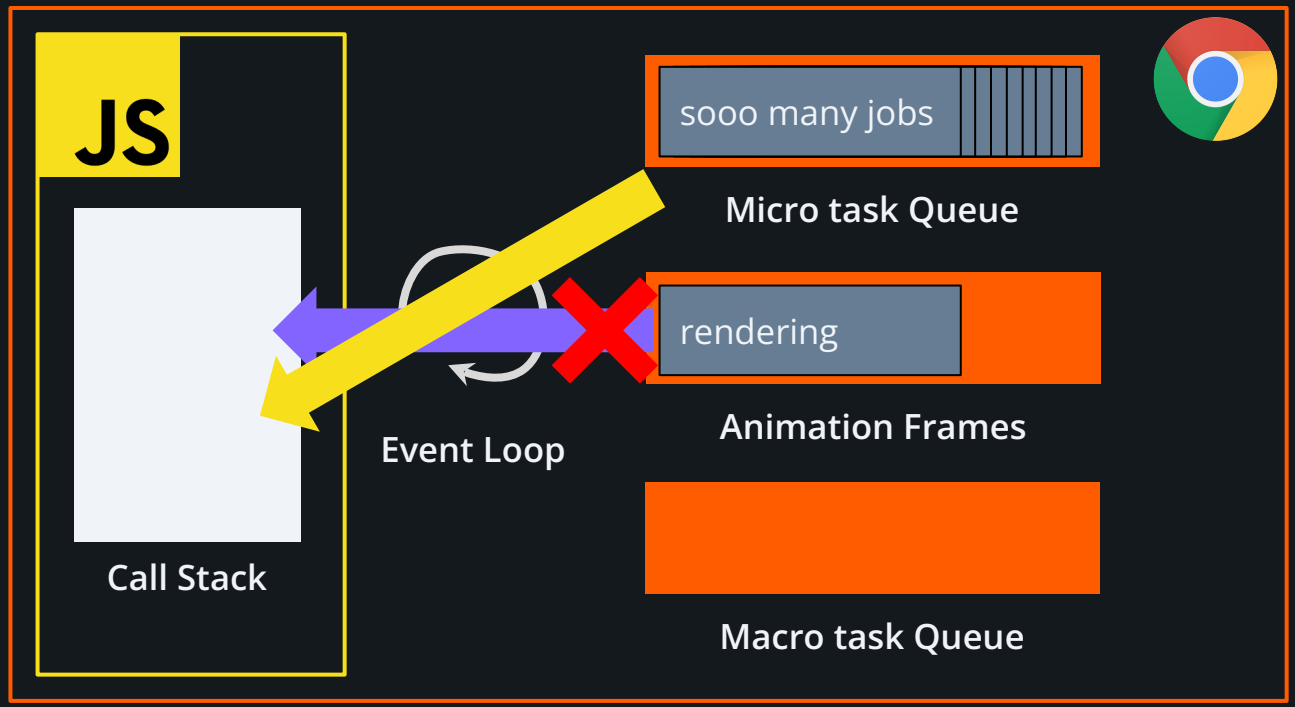


Event Loop란 무엇인가

우선순위

Macro Task Queue < Animation Frames < Micro task Queue

Event Loop란 무엇인가



wanted

6. Engineer가 가져야 하는 습관

Engineer가 가져야 하는 습관

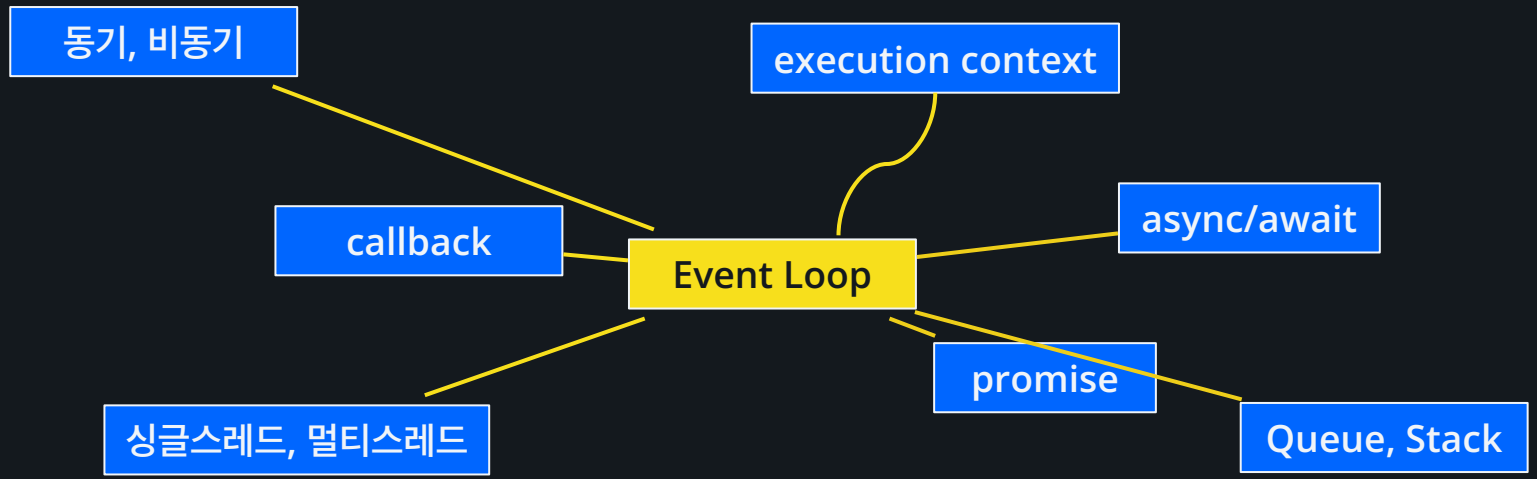
❓ 기술 면접 질문은 왜 하는가?

❓ 기업은 어떠한 사람을 원하는가?

❓ 기능 구현 보다 중요한 것은 무엇인가?

❗ **Engineer는 무엇에 집중해야 하는가?**

Engineer가 가져야 하는 습관



면접 단골 질문 이벤트 루프 다시 돌아보기

Engineer가 가져야 하는 습관

추상화 된 기능들에 관심을 가져야 한다

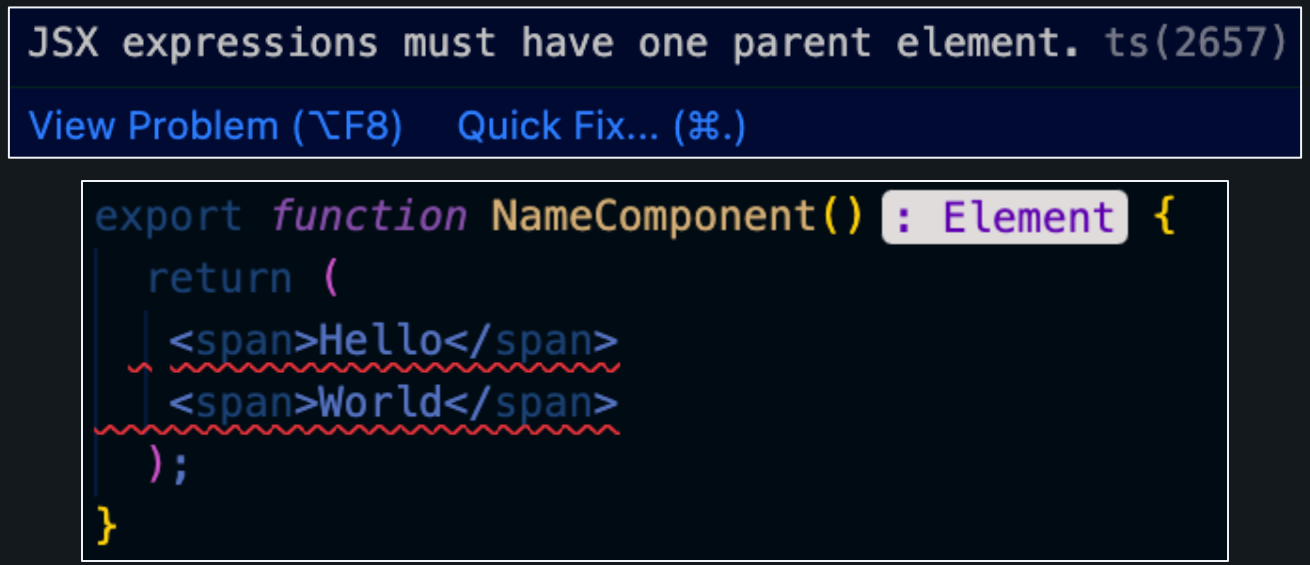
당연하게 동작하는 것은 없다

당연하게 생각하지 않기


export function NameComponent() : Element {
return (
<div style={{display: 'flex'}}>
황주현

);
}
JSX

당연하게 생각하지 않기


JSX expressions must have one parent element. ts(2657)
View Problem (⌥F8)
Quick Fix... (⌘.)
export function NameComponent() : Element {
return (
Hello
World
);
}

당연하게 생각하지 않기

```tsx
export function NameComponent() : Element {
  return (
    <>
      <span>Hello</span>
      <span>World</span>
    </>
  );
}
```

당연하게 생각하지 않기

```
// 변환 전
root.render(<h1>hello</h1>);
```


```
// 변환 후
root.render(createElement('h1', 'hello'));
```

당연하게 생각하지 않기

```
import {createElement} from 'react';  4.1k (gzipped: 1.8k)

export function NameComponent() : ReactElement<'hello', string | JSXElement… {
  return (createElement('span', 'hello') createElement('span','world'))
}
```

당연하게 생각하지 않기

```tsx
export function NameComponent() : Element {
  return (
    <>
      <span>Hello</span>
      <span>World</span>
    </>
  );
```

```tsx
import {createElement} from 'react';  4.1k (gzipped: 1.8k)

export function NameComponent() : (ReactElement<'hello', string | JSXElemen… {
  return [createElement('span', 'hello'), createElement('span', 'word')];
}
```

당연하게 생각하지 않기

```
createElement('div', [
  createElement('span', 'hello'),
  createElement('span', 'word'),
]);
```

당연하게 생각하지 않기

```
778  /**
779   * Create and return a new ReactElement of the given type.
780   * See https://reactjs.org/docs/react-api.html#createelement
781   */
782
783  function createElement(type, config, children) {
784    var propName; // Reserved names are extracted
785
786    var props = {};
787    var key = null;
788    var ref = null;
789    var self = null;
790    var source = null;
791
```

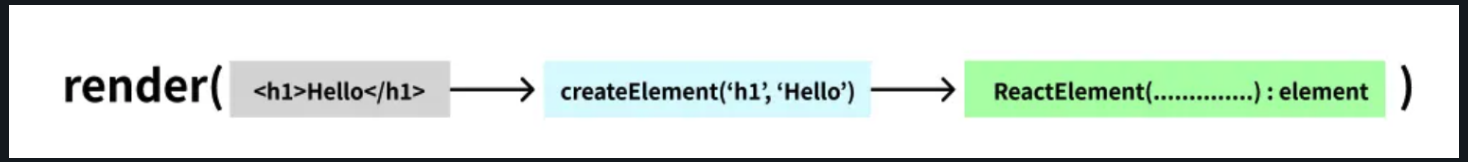
```
      }
    }

    return ReactElement(type, key, ref, self, source, ReactCurrentOwner.current, props);
  }
```

당연하게 생각하지 않기

```
var ReactElement = function (type, key, ref, self, source, owner, props) {
  var element = {
    // This tag allows us to uniquely identify this as a React Element
    $$typeof: REACT_ELEMENT_TYPE,
    // Built-in properties that belong on the element
    type: type,
    key: key,
    ref: ref,
    props: props,
    // Record the component responsi
    _owner: owner
  };
```

```
    return element;
};
```

당연하게 생각하지 않기


render(
<h1>Hello</h1>
createElement('h1', 'Hello')
ReactElement(.....) : element
)

Engineer가 가져야 하는 습관

모든 선택에 이유가 있어야 한다

그냥, 유명해서, 많이 사용해서 만으로는 이유가 되지 않는다

Engineer가 가져야 하는 습관

React를 왜 사용 하셨나요?

사람들이 많이 사용해서요! 🤓

Engineer가 가져야 하는 습관

React를 왜 사용 하셨나요?

프로젝트 팀원 구성이 모두 저연차이기 때문에
최대한 러닝커브를 줄이기 위해 사용량이 많고 커뮤니티가 활성화 되어 있는 React를 선택했습니다

Engineer가 가져야 하는 습관

Redux(상태 관리 라이브러리)를 왜 사용하셨나요?

팀장님이 쓰라던데요 😅

Engineer가 가져야 하는 습관

Redux(상태 관리 라이브러리)를 왜 사용하셨나요?

팀원들과 상태 관리 라이브러리 도입에 대해 이야기를 나누었고,
현재 저희 서비스 구조 상 많은 화면에서 Client State를 사용하고 있기 때문에 이를 효과적으로 관리하기 위해 사용했습니다.

Engineer가 가져야 하는 습관

도구 사용자

Engineer

wanted

고생하셨습니다.