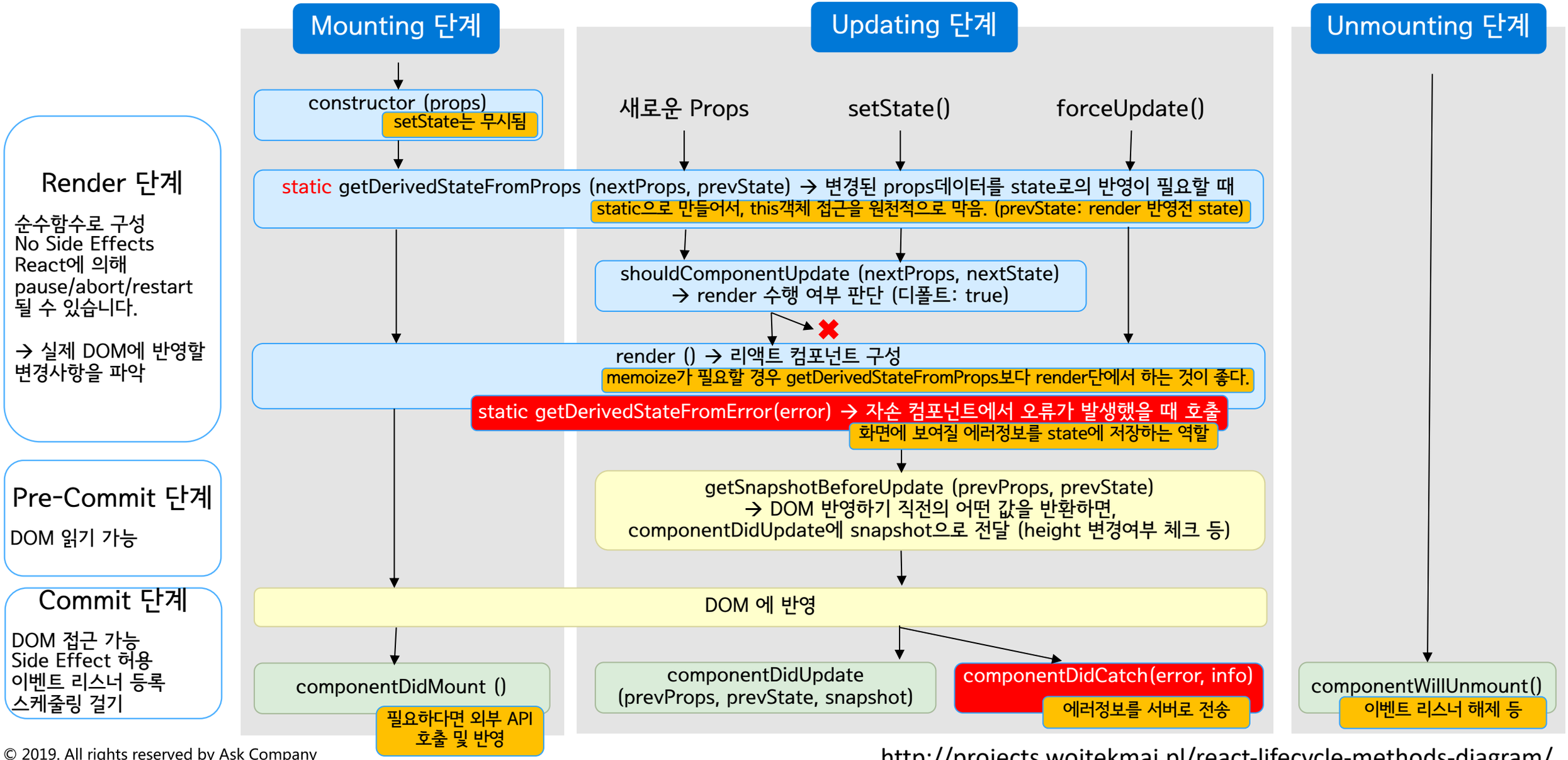
Ask Company

리액트와 함께 장고 시작하기 / 리액트

클래스 컴포넌트, 에러 처리

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

클래스 컴포넌트, 생명 주기



http://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/

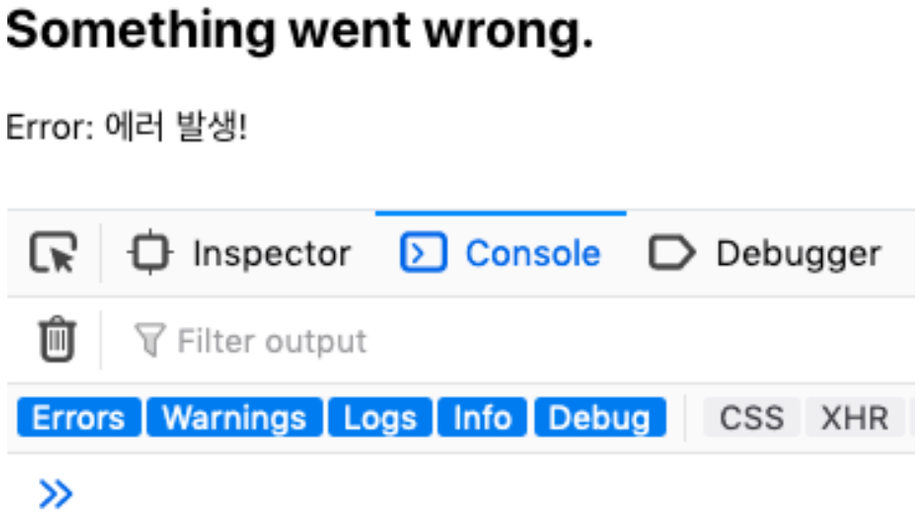
static getDerivedStateFromError (error) (1/3)

오류 정보를 화면에 보여주기 위한 state를 생성할 목적

children 컴포넌트의 생명주기 함수에서 오류가 발생했을 때에만 호출됨.

Event Listener에서 발생한 오류에 대해서는 호출되지 않습니다.

Render 단계이기에 Side Effects 불가.



https://ko.reactjs.org/docs/error-boundaries.html

https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Global_Objects/Error

static getDerivedStateFromError (error) (2/3)

```
class ErrorBoundary extends React.Component {
  state = { error: null };

  static getDerivedStateFromError(error) {
    return { error };
  }

  render() {
    const { error } = this.state;
    if ( error !== null ) {
      return (
        <div>
          <h2>Something went wrong.</h2>
          <div>{error.toString()}</div>
        </div>
      );
    }
    else
      return this.props.children;
  }
}
```

반환된 객체가 상탯값에 반영됩니다.

```
class Message extends React.Component {
  render() {
    throw new Error("에러 발생!");
    return (
      <div>Message Component</div>
    );
  }
}
```

강제로 예외를 발생시켜봅니다.

```
class App extends React.Component {
  render() {
    return (
      <div>
        <ErrorBoundary>
          <Message />
        </ErrorBoundary>
      </div>
    );
  }
}
```

static getDerivedStateFromError (error) (3/3)

개발서버에서는 별도 오류화면이 보여지기에,

커스텀 화면을 보기위해서는 production 빌드가 필요

```
yarn build --production # ./build/ 경로에 생성

# 1) nodejs의 serve 팩키지 활용하여 정적 서빙
yarn global add serve
serve -s ./build/

# 2) 파이썬의 http.server 모듈을 활용하여 정적 서빙
python -m http.server --directory ./build/
```

componentDidCatch (error, errorInfo) (1/2)

getDerivedStateFromError와 유사하게,

children 컴포넌트의 생명주기 함수에서 오류가 발생했을 때에만 호출.

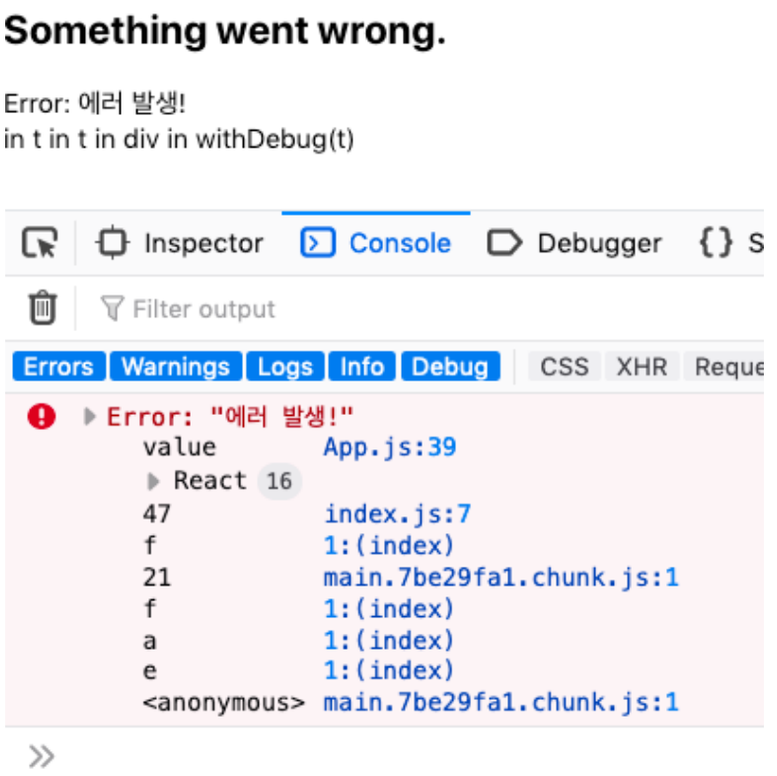
커밋 단계

→ 에러정보를 서버로의 전송이 가능 (Side effects 허용)

getDerivedStateFromError과 비교해서 errorInfo 정보가 추가로 전달

현재 setState이 호출이 가능하긴 하나, 향후에 막힐 가능성.

→ setState 호출은 getDerivedStateFromError에서만 수행합시다.



componentDidCatch (error, errorInfo) (2/2)

```
class ErrorBoundary extends React.Component {
  state = { error: null };

  static getDerivedStateFromError(error) {
    return { error };
  }

 componentDidCatch(error, errorInfo) {
    // FIXME: 향후에 막힌다고 합니다.
    this.setState({ error });
    // TODO: setState는 하지 마시고, 서버로 에러를 전달해볼 수 있겠습니다.
  }

  render() {
    const { error } = this.state;
    if ( error !== null ) {
      return (
        <div>
          <h2>Something went wrong.</h2>
          <div>{error.toString()}</div>
        </div>
      );
    }
    else
      return this.props.children;
  }
}
```

```
class Message extends React.Component {
  render() {
    throw new Error("에러 발생!");
    return (
      <div>Message Component</div>
    );
  }
}
```

강제로 예외를 발생시켜봅니다.

```
class App extends React.Component {
  render() {
    return (
      <div>
        <ErrorBoundary>
          <Message />
        </ErrorBoundary>
      </div>
    );
  }
}
```

Life is short.
You need Python and Django.

I will be your pacemaker.