addTransitionType

Canary

`addTransitionType` API는 현재 React의 카나리 및 실험적 채널에서만 사용할 수 있습니다.

React의 배포 채널에 대해 더 알아보세요.

`addTransitionType`은 트랜지션의 원인을 명시할 수 있습니다.

```
startTransition(() => {
  addTransitionType('my-transition-type');
  setState(newState);
});
```


레퍼런스

addTransitionType

매개변수

- `type` : 추가할 트랜지션의 타입입니다. 어떤 문자열이든 될 수 있습니다.

반환값

`startTransition` 은 아무것도 반환하지 않습니다.

주의 사항

- 여러 트랜지션이 결합되면 모든 트랜지션 타입이 수집됩니다. 하나의 트랜지션에 두 개 이상의 타입을 추가할 수도 있습니다.
- 트랜지션 타입은 커밋마다 초기화됩니다. 즉, `<Suspense>` 의 Fallback은 `startTransition` 이후 타입을 연결하며, 내용이 나타날 때는 그렇지 않습니다.

사용법

트랜지션의 원인 추가하기

트랜지션의 원인을 나타내기 위해 `startTransition` 내부에서 `addTransitionType` 을 호출합니다

```
import { startTransition, addTransitionType } from 'react';

function Submit({action) {
  function handleClick() {
    startTransition(() => {
      addTransitionType('submit-click');
      action();
    });
  }

  return <button onClick={handleClick}>Click me</button>;
}
```

`addTransitionType`을 `startTransition`의 범위 내에서 호출하면, React는 해당 트랜지션에 `submit-click`을 원인으로 연결합니다.

현재 트랜지션 타입은 원인에 따라 서로 다른 애니메이션을 커스터마이즈하는 데 사용할 수 있습니다. 사용할 수 있는 방식은 세 가지입니다.

- 브라우저 View Transition 타입으로 애니메이션 커스텀하기
- `View Transition` 클래스로 애니메이션 커스텀하기
- `ViewTransition` 이벤트로 애니메이션 커스텀하기

향후에는 트랜지션의 원인을 활용할 수 있는 다양한 용례를 지원할 예정입니다.

브라우저 View Transition 타입으로 애니메이션 커스텀하기

트랜지션에서 `ViewTransition`이 활성화되면, React는 모든 트랜지션 타입을 브라우저의 View Transition Types으로 해당 요소에 추가합니다.

이렇게 하면 CSS 범위에서 다른 애니메이션을 커스텀할 수 있습니다.

```
function Component() {
  return (
    <ViewTransition>
      <div>Hello</div>
    </ViewTransition>
  );
}

startTransition(() => {
  addTransitionType('my-transition-type');
  setShow(true);
});
```

```
:root:active-view-transition-type(my-transition-type) {
  &::view-transition-...(...) {
    ...
  }
}
```

`View Transition` 클래스로 애니메이션 커스텀하기

활성화된 `ViewTransition` 에서 타입에 따라 애니메이션을 커스터마이즈하려면, View Transition 클래스에 객체를 전달하면 됩니다.

```
function Component() {
  return (
    <ViewTransition enter={{
      'my-transition-type': 'my-transition-class',
    }}>
      <div>Hello</div>
    </ViewTransition>
  );
}

// ...
startTransition(() => {
  addTransitionType('my-transition-type');
  setState(newState);
});
```

여러 타입이 매칭되면 값들이 결합됩니다. 매칭되는 타입이 없으면 “default” 엔트리가 사용됩니다. 어떤 타입이라도 값이 “none”이면 해당 값이 우선하며 `ViewTransition` 은 비활성화됩니다. (이름이 할당되지 않습니다).

이 방식은 enter/exit/update/layout/share Props와 결합하여 트리거 종류와 트랜지션 타입에 따라 동작을 맞출 수 있습니다.

```
<ViewTransition enter={{
  'navigation-back': 'enter-right',
  'navigation-forward': 'enter-left',
}}
exit={{
  'navigation-back': 'exit-right',
  'navigation-forward': 'exit-left',
}}>
```

`ViewTransition` 이벤트로 애니메이션 커스텀하기

View Transition 이벤트를 활용하여 타입에 따라 활성화된 `ViewTransition` 의 애니메이션을 즉시 커스터마이즈할 수 있습니다.

```
<ViewTransition onUpdate={(inst, types) => {
  if (types.includes('navigation-back')) {
    ...
  } else if (types.includes('navigation-forward')) {
    ...
  } else {
    ...
  }
}}>
```

이를 통해 원인에 따라 다른 명령형 애니메이션을 선택할 수 있습니다.

이전
act

다음
cache

커뮤니티

행동 강령

팀 소개

문서 기여자

감사의 말

더 보기

블로그

React Native

개인 정보 보호

약관