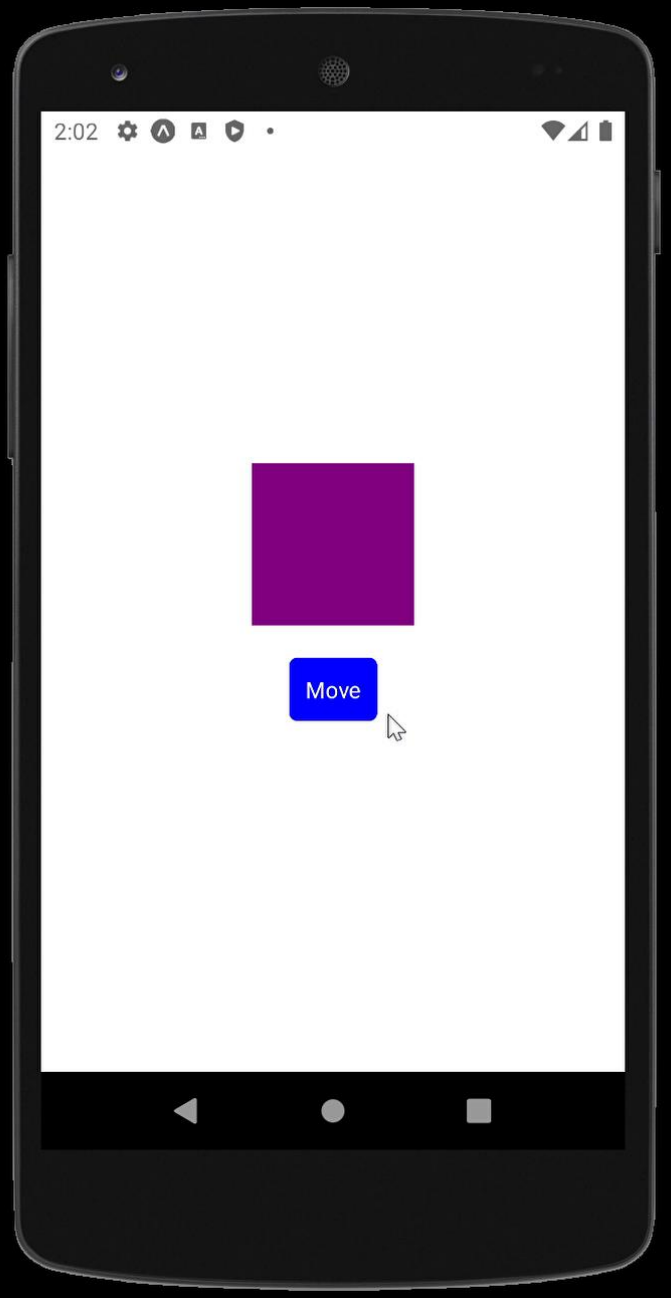
react-native 애니메이션

2024-01-02

```jsx
import React, { useRef } from 'react';
import { View, StyleSheet, Animated, Dimensions, TouchableOpacity, Text } from
'react-native';

// 로직
export default function App() {
  const position = useRef(new Animated.ValueXY()).current;
  const screenWidth = Dimensions.get('window').width;
  const screenHeight = Dimensions.get('window').height;


  //왼쪽 상단으로 이동하는 함수
  const moveToLeftTop = () => {
    Animated.timing(position, {
      toValue: { x: -screenWidth / 4, y: -screenHeight / 4 }, // 화면 왼쪽 상단으로 이
동
      duration: 1000, // 애니메이션 소요 시간 (밀리초)
      useNativeDriver: false,
    }).start(() => {
      // 애니메이션이 완료되면 중앙으로 초기화
      Animated.timing(position, {
        toValue: { x: 0, y: 0 },
        duration: 0,
        useNativeDriver: false,
      }).start();
    });
  };



  // 화면
    return (
    <View style={styles.container}>
      <Animated.View style={[styles.box, { transform:
position.getTranslateTransform() }]} />
      <TouchableOpacity onPress={moveToLeftTop}>
        <View style={styles.button}>
          <Text style={styles.buttonText}>Move</Text>
        </View>
      </TouchableOpacity>
    </View>
  );
}
```

```jsx
// css
const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
  box: {
    width: 100,
    height: 100,
    backgroundColor: 'purple',
  },
  button: {
    marginTop: 20,
    padding: 10,
    backgroundColor: 'blue',
    borderRadius: 5,
  },
  buttonText: {
    color: 'white',
  },
});
```



1. import

```
import React, { useRef } from 'react';
import { View, StyleSheet, Animated, Dimensions, TouchableOpacity, Text } from 'react-native';
```

1. useRef

   - useRef란 원하는 특정 DOM을 직접 선택해서 컨트롤 할 수 있게 해주는 Hook
   - useRef()를 쓰면 새롭게 렌더링 되지 않고 값이 유지 되게 된다.
   - 애니메이션이 되고 원래 위치로 돌아가는 것을 방지하는 데 사용한다.
   - 즉, 리렌더링 시 값이 초기화 되는 것을 막기 위해서 사용한다.

   공식 문서 : https://react.dev/reference/react/useRef

   https://qcoding.tistory.com/14 https://jiiiihong.space/entry/useRef-%EB%9E%80

2. Animated

   공식 문서 : https://reactnative.dev/docs/animations

참고 :
- Hook이해를 위한 useState의 간단한 예시 https://thinkmath2020.tistory.com/3536
- react native DOM
  - 파싱 : 웹페이지에서 원하는 데이터를 추출하여 가공하기 쉬운 상태로 바꾸는 것
  - 렌더링 : 파싱의 과정
  - DOM (문서객체) : 렌더링 과정에서 만들어진 자바스크립트 객체
  - https://velog.io/@heejin62/React-Native-%ED%94%84%EB%A0%88%EC%9E%84%EC%9B%8C%ED%81%AC-%EC%9E%91%EB%8F%99-%EC%9B%90%EB%A6%AC-%EA%B0%80%EC%83%81-DOM%EA%B3%BC-%EB%A0%8C%EB%8D%94%EB%A7%81

2. 로직 (1)

```
// 로직
export default function App() {
  const position = useRef(new Animated.ValueXY()).current;
  const screenWidth = Dimensions.get('window').width;
  const screenHeight = Dimensions.get('window').height;


  //왼쪽 상단으로 이동하는 함수
  const moveToLeftTop = () => {
    Animated.timing(position, {
      toValue: { x: -screenWidth / 4, y: -screenHeight / 4 }, // 화면 왼쪽 상단으로 이동
      duration: 1000, // 애니메이션 소요 시간 (밀리초)
      useNativeDriver: false,
    }).start(() => {
      // 애니메이션이 완료되면 중앙으로 초기화
      Animated.timing(position, {
        toValue: { x: 0, y: 0 },
        duration: 0,
        useNativeDriver: false,
      }).start();
    });
  };
```

1. Animated.ValueXY
   - x축,y축 값을 가짐

2. .current
   - 실제 값에 접근하기 위해 사용

공식 문서 :

https://reactnative.dev/docs/animatedvaluexy

2. 로직 (2)

```
// 로직
export default function App() {
  const position = useRef(new Animated.ValueXY()).current;
  const screenWidth = Dimensions.get('window').width;
  const screenHeight = Dimensions.get('window').height;


  //왼쪽 상단으로 이동하는 함수
  const moveToLeftTop = () => {
    Animated.timing(position, {
      toValue: { x: -screenWidth / 4, y: -screenHeight / 4 }, // 화면 왼쪽 상단으로 이동
      duration: 1000, // 애니메이션 소요 시간 (밀리초)
      useNativeDriver: false,
    }).start(() => {
      // 애니메이션이 완료되면 중앙으로 초기화
      Animated.timing(position, {
        toValue: { x: 0, y: 0 },
        duration: 0,
        useNativeDriver: false,
      }).start();
    });
  };
```

2. 로직 (2)_ timing() & start()

timing() https://reactnative.dev/docs/animated

`timing()`

```
static timing(value, config): CompositeAnimation;
```

시간이 지정된 이징 곡선을 따라 값에 애니메이션을 적용합니다. 이 `Easing` 모듈에는 사전 정의된 수많은 곡선이 있거나 자신만의 기능을 사용할 수 있습니다.

Config는 다음과 같은 옵션을 가질 수 있는 개체입니다.

- `duration`: 애니메이션 길이(밀리초)입니다. 기본값은 500입니다.
- `easing`: 곡선을 정의하는 이징 기능입니다. 기본값은 입니다 `Easing.inOut(Easing.ease)`.
- `delay`: 지연(밀리초) 후에 애니메이션을 시작합니다. 기본값은 0입니다.
- `isInteraction`: 이 애니메이션이 에 "상호작용 핸들"을 생성하는지 여부입니다 `InteractionManager`. 기본값은 true입니다.
- `useNativeDriver`: true인 경우 기본 드라이버를 사용합니다. 필수의.

```
Animated.timing(animation, {
  toValue: 0, // 어떤 값으로 변경할지 - 필수
  duration: 1000, // 애니메이션에 걸리는 시간(밀리세컨드) - 기본값 500
  delay: 0, // 딜레이 이후 애니메이션 시작 - 기본값 0
  useNativeDriver: true, // 네이티브 드라이버 사용 여부 - 필수
  isInteraction: true, // 사용자 인터랙션에 의해 시작한 애니메이션인지 지정 - 기본값 true
  easing: Easing.inOut(Easing.ease), // 애니메이션 속서 변경 함수 - 기본값 Easing.inOut(Easing.ease)
}).start(() => {
  // 애니메이션 처리 완료 후 실행할 작업
})
```

https://devbksheen.tistory.com/entry/Animated%EB%A1%9C-%EC%95%A0%EB%8B%88%EB%A9%94%EC%9D%B4%EC%85%98-%EC%A0%81%EC%9A%A9%ED%95%98%EA%B8%B0

start()

`start()`

```
static start(callback?: (result: {finished: boolean}) => void);
```

애니메이션은 애니메이션에서 start()를 호출하여 시작됩니다. start()는 애니메이션이 완료되거나 애니메이션이 완료되기 전에 stop()이 호출되었기 때문에 애니메이션이 완료될 때 호출되는 완료 콜백을 사용합니다.

매개변수:

이름	유형	필수의	설명
콜백	`(result: {finished: boolean}) => void`	아니요	애니메이션이 정상적으로 실행된 후 또는 애니메이션이 완료되기 전에 stop()이 호출되어 애니메이션이 완료된 경우 호출되는 함수

3. 화면

```
// 화면
  return (
  <View style={styles.container}>
    <Animated.View style={[styles.box, { transform: position.getTranslateTransform() }]} />
    <TouchableOpacity onPress={moveToLeftTop}>
      <View style={styles.button}>
        <Text style={styles.buttonText}>Move</Text>
      </View>
    </TouchableOpacity>
  </View>
  );
}
```

1.<Animated.View style={[styles.box, { transform: position.getTranslateTransform() }]}/>

- .getTranslateTransform() : position의 애니메이션 값을 이동 행렬로 변환하여 적용
- 애니메이션에 따라 이동하는 효과, 뷰에 반영
- react native Transforms 관련 공식 문서 : https://reactnative.dev/docs/transforms

2. <TouchableOpacity onPress={moveToLeftTop}>

- '로직 (2)'의 moveToLeftTop함수 실행

※ 애니메이션의 시작 및 완료는 비동기적으로 처리되므로

<Animated.View>가 먼저 렌더링되더라도 실제 애니메이션이 발생하는 시점은 사용자가 버튼을 누를 때

참고 사이트

- react-native 속성 정리 사이트 : https://www.codedaily.io/courses/React-Native-Animated-for-Beginners/Animatedtiming
- react 강의 : https://react-ko.dev/learn/referencing-values-with-refs