Version: 3.x

# useHandler

> INFO
>
> This page was ported from an old version of the documentation.
>
> As we're rewriting the documentation some of the pages might be a little outdated.

This is low-level hook returning context object and value indicating whether worklet should be rebuilt, which should be used in order to create custom event handler hook like `useAnimatedGestureHandler` or `useAnimatedScrollHandler`.

## Arguments

### `handlerOrHandlersObject` **[object with worklets]**

Object containing custom keys matching native event names. The values in the object should be individual worklets. Each of the worklet will be triggered when the corresponding event is dispatched on the connected animated component.

Each of the event worklets will receive the following parameters when called:

- `event` [object] - event object. The payload can differ depending on the type of the event.
- `context` [object] - plain JS object that can be used to store some state. This object will persist in between event occurrences and you can read and write any data to it. When there are several event handlers provided in a form of an object of worklets, the `context` object will be shared in between the worklets allowing them to communicate with each other.

### `dependencies` **[Array]**

Optional array of values which changes cause this hook to receive updated values during rerender of the wrapping component. This matters when, for instance, worklet uses values dependent on the component's state.

`dependencies` here may be:

- `undefined` (argument skipped) - worklet will be rebuilt if there is any change in any of the callbacks' bodies or any values from their closure(variables from outer scope used in worklet),
- empty array( `[]` ) - worklet will be rebuilt only if any of the callbacks' bodies changes,
- array of values( `[val1, val2, ..., valN]` ) - worklet will be rebuilt if there is any change in any of the callbacks bodies or in any values from the given array.

### Returns

The hook returns a context that will be reused by event handlers and value that indicates whether worklets should be rebuilt. If different implementation is needed for web, `useWeb` boolean is returned to check for web environment

## Example

```
function useAnimatedPagerScrollHandler(handlers, dependencies) {
  const { context, doDependenciesDiffer, useWeb } = useHandler(handlers,
dependencies);

  return useEvent(
    (event) => {
      'worklet';
      const { onPageScroll } = handlers;

      if (onPageScroll && event.eventName.endsWith('onPageScroll')) {
        onPageScroll(event, context);
      }
    },
    ['onPageScroll'],
    doDependenciesDiffer,
  );
```

Edit this page