 alexreardon fixing outdated references to data attributes

2 contributors 

229 lines (194 sloc) | 5.81 KB

Types

`react-beautiful-dnd` is typed using `flowtype`. This greatly improves internal consistency within the codebase. We also expose a number of public types which will allow you to type your javascript if you would like to. If you are not using `flowtype` this will not inhibit you from using the library. It is just extra safety for those who want it.

Public flow types

Ids

```
type Id = string;
type TypeId = Id;
type DroppableId = Id;
type DraggableId = Id;
```

Responders

```
type Responders = {
  // optional
  onBeforeCapture?: OnBeforeCaptureResponder,
  onBeforeDragStart?: OnBeforeDragStartResponder,
  onDragStart?: OnDragStartResponder,
  onDragUpdate?: OnDragUpdateResponder,
```

```
  // required
  onDragEnd: OnDragEndResponder,
|};

type OnBeforeCaptureResponder = (before: BeforeCapture) => mixed;
type OnBeforeDragStartResponder = (start: DragStart) => mixed;
type OnDragStartResponder = (
  start: DragStart,
  provided: ResponderProvided,
) => mixed;
type OnDragUpdateResponder = (
  update: DragUpdate,
  provided: ResponderProvided,
) => mixed;
type OnDragEndResponder = (
  result: DropResult,
  provided: ResponderProvided,
) => mixed;

type BeforeCapture = {|
  draggableId: DraggableId,
  mode: MovementMode,
|};

type DraggableRubric = {|
  draggableId: DraggableId,
  type: TypeId,
  source: DraggableLocation,
|};

type DragStart = {|
  ...DraggableRubric,
  mode: MovementMode,
|};

type DragUpdate = {|
  ...DragStart,
  // populated if in a reorder position
  destination: ?DraggableLocation,
  // populated if combining with another draggable
  combine: ?Combine,
|};

// details about the draggable that is being combined with
type Combine = {|
  draggableId: DraggableId,
  droppableId: DroppableId,
|};

type DropResult = {|
  ...DragUpdate,
  reason: DropReason,
|};
```

```
type DropReason = 'DROP' | 'CANCEL';

type DraggableLocation = {|
  droppableId: DroppableId,
  // the position of the droppable within a droppable
  index: number,
|};

// There are two modes that a drag can be in
// FLUID: everything is done in response to highly granular input (eg mouse)
// SNAP: items snap between positions (eg keyboard);
type MovementMode = 'FLUID' | 'SNAP';
```

Sensors

```
type Sensor = (api: SensorAPI) => void;
type SensorAPI = {|
  tryGetLock: TryGetLock,
  canGetLock: (id: DraggableId) => boolean,
  isLockClaimed: () => boolean,
  tryReleaseLock: () => void,
  findClosestDraggableId: (event: Event) => ?DraggableId,
  findOptionsForDraggable: (id: DraggableId) => ?DraggableOptions,
|};
type TryGetLock = (
  draggableId: DraggableId,
  forceStop?: () => void,
  options?: TryGetLockOptions,
) => ?PreDragActions;
type TryGetLockOptions = {
  sourceEvent?: Event,
};
```

Droppable

```
type DroppableProvided = {|
  innerRef: (?HTMLElement) => void,
  placeholder: ?ReactElement,
  droppableProps: DroppableProps,
|};
type DroppableProps = {|
  // used for shared global styles
  'data-rbd-droppable-context-id': ContextId,
  // Used to lookup. Currently not used for drag and drop lifecycle
  'data-rbd-droppable-id': DroppableId,
|};
type DroppableStateSnapshot = {|
  isDraggingOver: boolean,
```

```
  draggingOverWith: ?DraggableId,
  draggingFromThisWith: ?DraggableId,
  isUsingPlaceholder: boolean,
|};
```

Draggable

```
type DraggableProvided = {|
  innerRef: (?HTMLElement) => void,
  draggableProps: DraggableProps,
  dragHandleProps: ?DragHandleProps,
|};

type DraggableStateSnapshot = {|
  isDragging: boolean,
  isDropAnimating: boolean,
  dropAnimation: ?DropAnimation,
  draggingOver: ?DroppableId,
  combineWith: ?DraggableId,
  combineTargetFor: ?DraggableId,
  mode: ?MovementMode,
|};

type DraggableProps = {|
  style: ?DraggableStyle,
  'data-rbd-draggable-context-id': string,
  'data-rbd-draggable-id': string,
  onTransitionEnd: ?(event: TransitionEvent) => void,
|};
type DraggableChildrenFn = (
  DraggableProvided,
  DraggableStateSnapshot,
  DraggableRubric,
) => Node | null;
|};
type DraggableStyle = DraggingStyle | NotDraggingStyle;
type DraggingStyle = {|
  position: 'fixed',
  top: number,
  left: number,
  boxSizing: 'border-box',
  width: number,
  height: number,
  transition: string,
  transform: ?string,
  zIndex: number,
  opacity: ?number,
  pointerEvents: 'none',
|};
type NotDraggingStyle = {|
  transition: ?string,
```

```
  transition: null | 'none',
|};

type DragHandleProps = {|
  onFocus: () => void,
  onBlur: () => void,
  onMouseDown: (event: MouseEvent) => void,
  onKeyDown: (event: KeyboardEvent) => void,
  onTouchStart: (event: TouchEvent) => void,
  tabIndex: number,
  'data-rbd-drag-handle-draggable-id': string,
  'data-rbd-drag-handle-context-id': string,
  role: string,
  'aria-describedby': string,
  draggable: boolean,
  onDragStart: (event: DragEvent) => void,
|};

type DropAnimation = {|
  duration: number,
  curve: string,
  moveTo: Position,
  opacity: ?number,
  scale: ?number,
|};
```

Using the flow types

The types are exported as part of the module so using them is as simple as:

```
import type { DroppableProvided } from 'react-beautiful-dnd';
```

Typescript

If you are using TypeScript you can use the community maintained DefinitelyTyped type definitions. Installation instructions.

Here is an example written in typescript.

Sample application with flow types

We have created a sample application which exercises the flowtypes. It is a super simple `React` project based on `react-create-app`. You can use this as a reference to see how to set things up correctly.

← Back to documentation