Connected React Components

```
type ListerContainerState = {
  activeNavigation: Array<NavCategory>,
};

const mapStateToProps = (state: AppState): ListerContainerState => ({
  activeNavigation: getMenuCategoryFromURL(state),
});

type OwnProps = {/* ... */};
type ReduxActions = typeof mapDispatchToProps;
type ReduxState = $Exact<ExtractReturn<typeof mapStateToProps>>;

export type Props = {|
  ...ReduxActions,
  ...ReduxState,
  ...OwnProps
|};
```

Connected React Components

```
type ListerContainerState = {
  activeNavigation: Array<NavCategory>,
};

const mapStateToProps = (state: AppState): ListerContainerState => ({
  activeNavigation: getMenuCategoryFromURL(state),
});

type OwnProps = {/* ... */};
type ReduxActions = typeof mapDispatchToProps;
type ReduxState = $Exact<ExtractReturn<typeof mapStateToProps>>;

export type Props = {|
  ...ReduxActions,
  ...ReduxState,
  ...OwnProps
|};
```

Connected React Components

```
type ListerContainerState = {
  activeNavigation: Array<NavCategory>,
};

const mapStateToProps = (state: AppState): ListerContainerState => ({
  activeNavigation: getMenuCategoryFromURL(state),
});

type OwnProps = {/* ... */};
type ReduxActions = typeof mapDispatchToProps;
type ReduxState = $Exact<ExtractReturn<typeof mapStateToProps>>;

export type Props = {|
  ...ReduxActions,
  ...ReduxState,
  ...OwnProps
|};
```

Connected React Components

```
type ListerContainerState = {
  activeNavigation: Array<NavCategory>,
};

const mapStateToProps = (state: AppState): ListerContainerState => ({
  activeNavigation: getMenuCategoryFromURL(state),
});

type OwnProps = {/* ... */};
type ReduxActions = typeof mapDispatchToProps;
type ReduxState = $Exact<ExtractReturn<typeof mapStateToProps>>;

export type Props = {|
  ...ReduxActions,
  ...ReduxState,
  ...OwnProps
|};
```

React Children

```
import * as React from 'react';

type Props = {
  children?: React.Node
};
```

React Components

```
import * as React from 'react';

class Lister extends React.Component<PropTypes, StateType> {
  /* ... */
}
```

React Higher order components

```
import * as React from 'react';

const HOC = (Component: React.ComponentType<*>) =>
  (props: PropTypes) =>
    <Component {...props} />;
```

Typing selectors

```
export const selectedVariantHasStockSelector: AppState => boolean = createSelector(
  selectedVariantSelector,
  (variant: ColorVariant) =>
    !!variant && variant.skus.reduce((acc, sku) => acc || hasStock(sku), false),
);
```

Typing selectors

```
export const selectedVariantHasStockSelector: AppState => boolean = createSelector(
  selectedVariantSelector,
  (variant: ColorVariant) =>
    !!variant && variant.skus.reduce((acc, sku) => acc || hasStock(sku), false),
);
```

Typing selectors

```
export const selectedVariantHasStockSelector: AppState => boolean = createSelector(
  selectedVariantSelector,
  (variant: ColorVariant) =>
    !!variant && variant.skus.reduce((acc, sku) => acc || hasStock(sku), false),
);
```

Redux immutable

```
type State = {
  +users: Array<{
    +id: string,
    +name: string,
    +age: number,
    +phoneNumber: string,
  }>,
  +activeUserID: string,
  // ...
};
```

Inference

```
declare function convert <A>(
  output: A,
  fn: (query: string) => A,
  fn2: (query: string) => A
): void;

// Expected errors, but didn't get
convert(
  1, // <- this value specifies type A to be number
  function(a: string) { return 1; },
  function(a: string) { return '1'; } // <- here it returns a string which is a wrong type, but flow didn't get it
);
```

Inference

```
declare function convert <A>(
  output: A,
  fn: (query: string) => A,
  fn2: (query: string) => A
): void;

// Expected errors, but didn't get
convert(
  1, // <- this value specifies type A to be number
  function(a: string) { return 1; },
  function(a: string) { return '1'; } // <- here it returns a string which is a wrong type, but flow didn't get it
);
```

Any vs Star

Any vs Star

- any performs type erasal

Any vs Star

- any performs type erasal
- star performs type inference

Editor integration

- Atom with Nuclide
- VScode

Command line

- `$ flow --show-all-branches`
- `$ flow-typed install`