TypeScript convention

Component

> ***Public components*** *are considered all components exported from* `@mui/material` *or* `@mui/lab` *.*
>
> ***Internal components*** *are considered all components that are not exported from the packages, but only used in some public component.*

`Props Interface`

- export interface `{ComponentName}classes` from `{component}Classes.ts` and add comment for generating api docs (for internal components, may or may not expose classes but don't need comment)
- export interface `{ComponentName}Props`
- always export props interface (use `interface` over `type`) from the component file

▶ Public component
▶ internal component

`ClassKey`

- naming as `{ComponentName}ClassKey`
- export if `classes` exists in props interface using `keyof` from `{component}Classes.ts`

```
// fooClasses.ts
export interface FooClasses {
  ...
}

export type FooClassKey = keyof FooClasses;
// verify that FooClassKey is union of string literal
```

`Classes generator & Utility`

- export if `classes` exists in props interface from the component file
- use `{Component}Classes` as type to preventing typo and missing classes
- use `Private` prefix for internal component

▶ Public component
▶ internal component

`StyledComponent`

- naming using slot `{ComponentName}{Slot}`
- to extend interface of the styled component, pass argument to generic

▶ public component
▶ internal component
▶ extends interface

`Component declaration`

- prefer `function Component() {}` over `React.FC`
- naming the render function in `React.forwardRef` (for devtools)
- `useThemeProps` is needed only for public component

- pass `ownerState` to StyledComponent for styling

▶ public component

▶ internal component