Exercises

Exercise 1: Theme variants

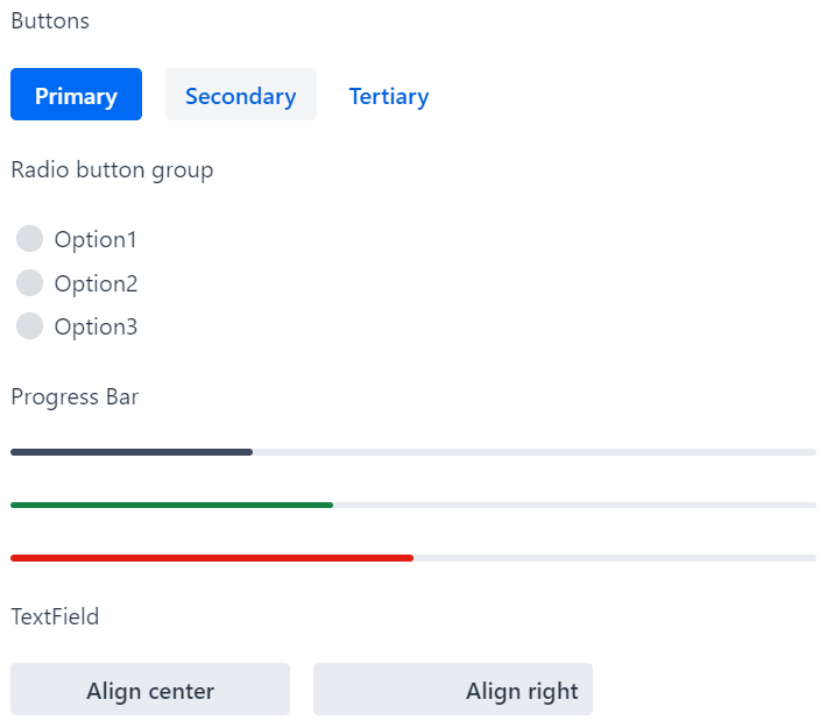
Theme variant is the easiest way to change a component's look and feel. For this exercise, you are supposed to apply different theme variants to some components. To apply a theme variant, just call `component.addThemeVariants();`

There are predefined Variants, e.g.,
`button.addThemeVariants(ButtonVariant.LUMO_PRIMARY);`

For this exercise, you need to:

1. Apply "primary", "secondary", "tertiary" to buttons
2. Apply "vertical" to a RadioButtonGroup
3. Apply "contrast", "success", "error" to progress bars
4. Apply "align-center", "align-right" to Textfields

The result should look like this:



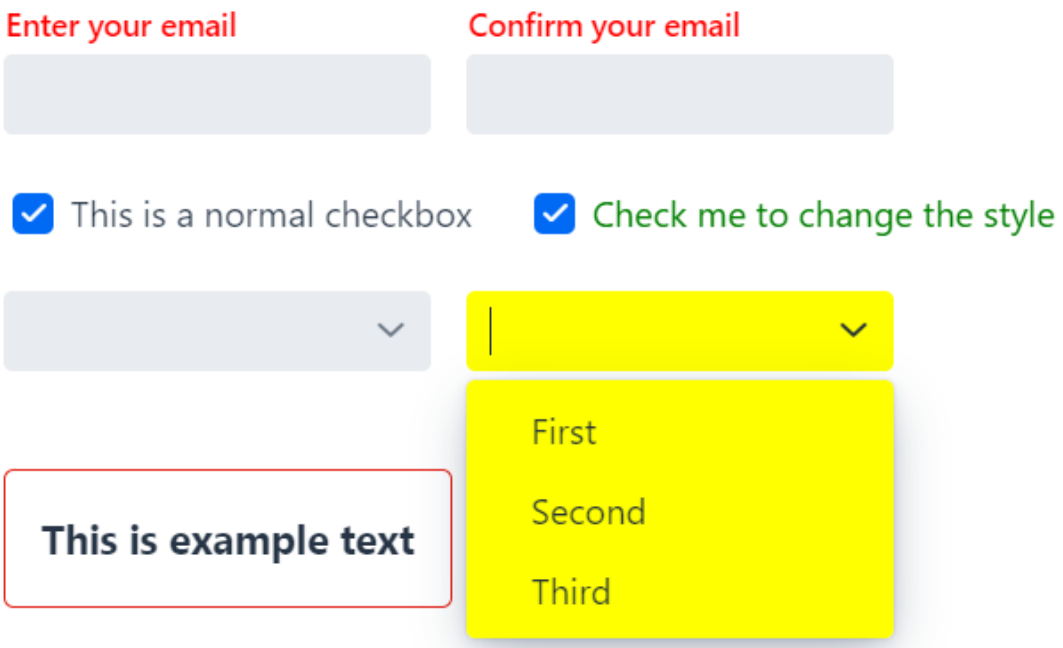
Exercise 2: Styling component parts and states

When you need to style a Vaadin component, and the Lumo CSS properties or Theme Variants are not sufficient, usually styling based on the component parts or states is the way to go.

For this exercise, you need to style several components and a Span with text.

1. Using only modifications in frontend/themes/training/styles.css, style the labels of all EmailFields in the application so that their labels are red. You can review the Email Field styling documentation to look for a CSS selector which will help you achieve that.
2. Style *only* the right checkbox in a way that it turns green when it's checked. Search the Checkbox styling documentation for a selector which could help you to do that. You'll first have to set some specific CSS class only to the right instance of the checkbox using the method `styledCheckbox.setClassName()`.
3. Search ComboBox styling documentation which could help you to color the right ComboBox background yellow, including the popup overlay. Note that you can use the `yellowBgColorCombo.setOverlayClassName` method to set a CSS class for the ComboBox overlay.
4. Using only the CSS classes provided by the LumoUtility Java class, style the Span text so it looks like in the picture below.

The result should look like this:



Exercise 3: Grid Dynamic Styling

The view has a TextField and a Grid. The grid contains a listing of expenses for 12 months. The TextField is for entering a monthly expense limit. If a month's expenses listed in the grid are higher than the expense limit defined in the TextField, then the expense cell's text should be highlighted with a red color.

You'll need to do the following things:

1. Set a part name generator to the Expenses column (setPartNameGenerator method).
2. In the generator, check if the monthly expenses exceed the limit; if yes, then return a "warn", otherwise return null.
3. Color the text red in the grid in frontend/themes/training/styles.css using the `vaadin-grid::part(warn)` selector

The result should look like this (the number of rows in red depends on the entered limit):

Limit for monthly expenses

500

Month	Expenses
JANUARY	300
FEBRUARY	350
MARCH	400
APRIL	450
MAY	500
JUNE	550
JULY	600
AUGUST	650
SEPTEMBER	700
OCTOBER	750
NOVEMBER	800
DECEMBER	850