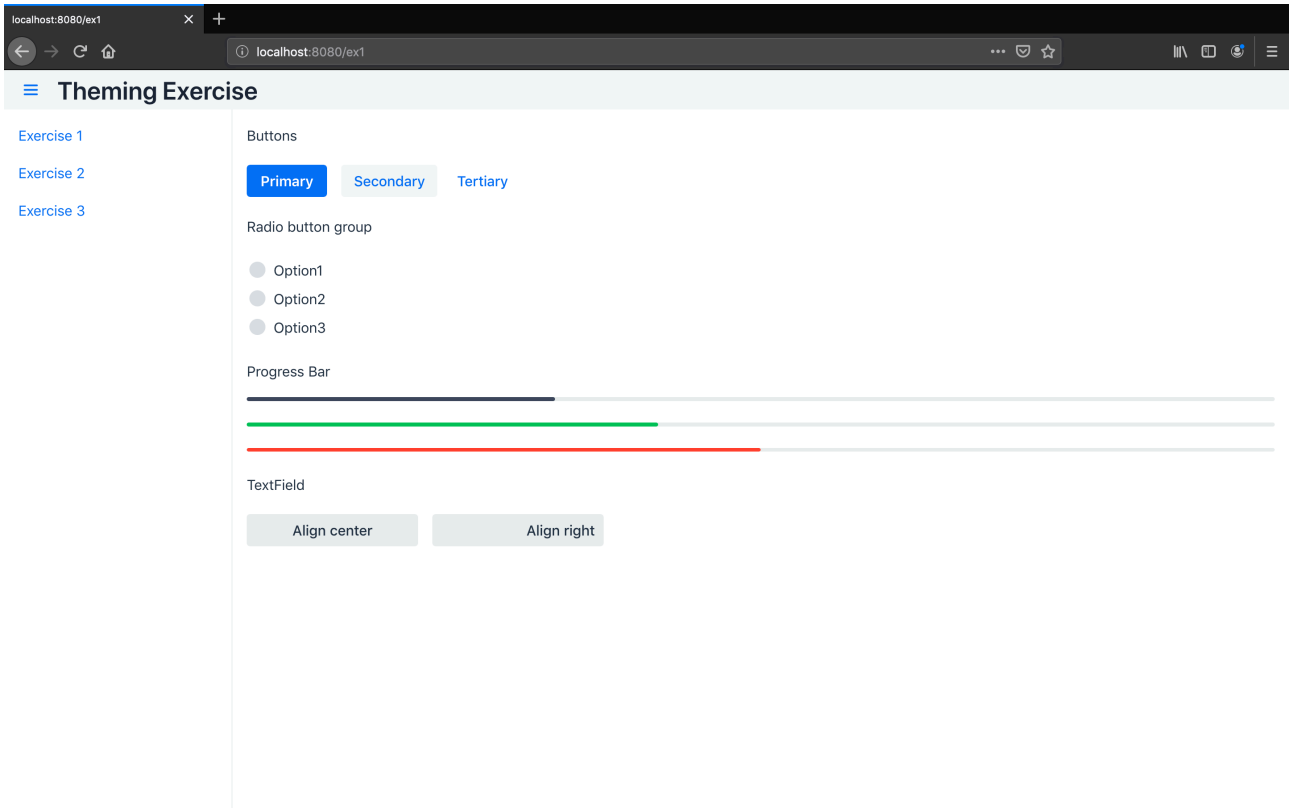
 

# Exercise 1: Theme variants

Theme variant is the easiest way to change a component's look and feel. For this exercise you are supposed to apply different theme variant to some components. To apply a theme variant, you need to use the Element API
`component.getElement().getThemeList().add("variant-name");`

From Vaadin 12, you will be able to just call
`component.addThemeVariants();`
`and There are predefined Variants, e.g.`
`button.addThemeVariants(ButtonVariant.LUMO_PRIMARY);`

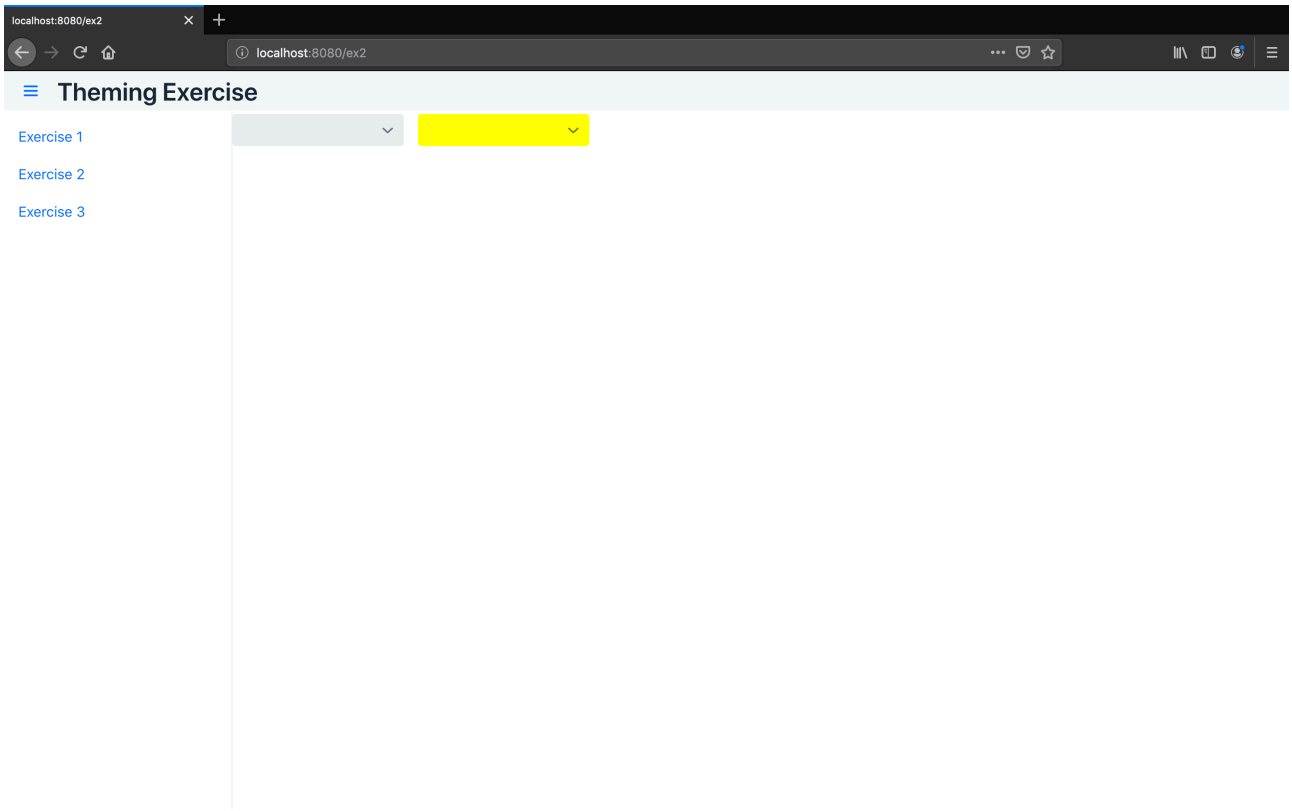



For this exercise, you need to:
1. Apply "primary", "secondary", "tertiary" to buttons
2. Apply "vertical" to a RadioButtonGroup
3. Apply "contrast", "success", "error" to progress bars
4. Apply "align-center", "align-right" to Textfields

# Exercise 2: Theme module

When you need to style a Vaadin component, usually theme module is the way to go. For this exercise, you need to style a combobox to have a yellow background.

Inside exercise 2 view, there are 2 comboboxes. Style the yellowBgColorCombo, so that the combobox has a yellow background color.



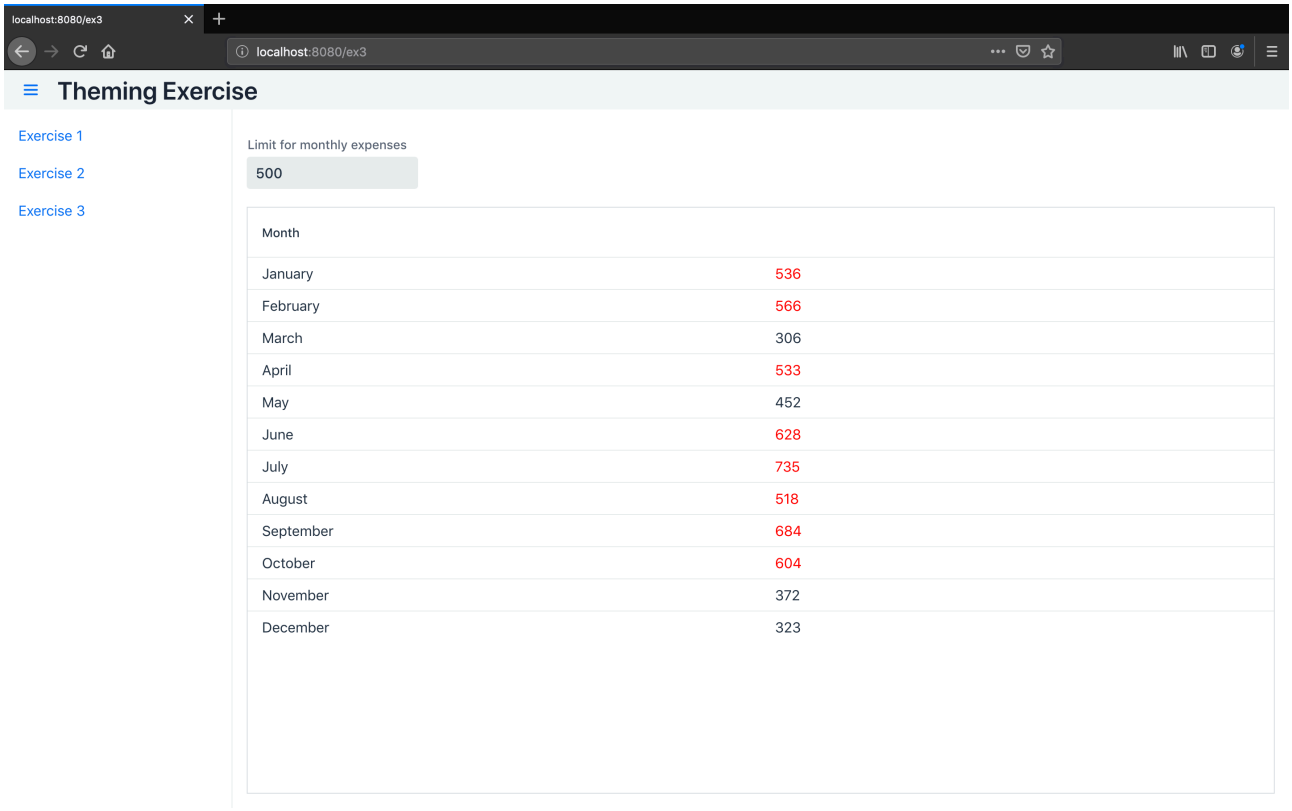

For this exercise, you need to:

1. After inspecting, you can find that combobox uses a vaadin-text-field, we actually want to make the text field to have a yellow background.
2. Update the theme attribute of the second combobox, so that the theme attribute and propagate to the vaadin-text-field in the combobox e.g. yellowBgColorCombo.getElement().getThemeList().add("**yellowBg**");
3. Make a CSS file for styling vaadin-text-field. e.g. vaadin-text-field-yellow-bg.css
4. In the MainLayout.java, import the CSS as a style module for vaadin-text-field. @CssImport(value = "./styles/vaadin-text-field-yellow-bg.css", themeFor = "vaadin-text-field")
5. In the CSS file, use :host([theme~=“**yellowBg**"]) to target a vaadin-text-field with “yellowBg” as theme attribute. Use [part=“input-field"] to target the “input-field” part
6. Apply background-color: yellow;

# 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 class name generator to the Expenses column.
2. In the generator, check if the monthly expenses exceeds the limit, if yes, then return a "warn", otherwise return null.
3. Make a CSS file for styling the grid. Use a class name selector .warn to make the color red.
4. Use @CssImport to import the CSS as style module for vaadin-grid in MainLayout.java.