Internationalization & Localization

[Design System Image]

[DemoCard Component - Interactive React component]

Internationalization and localization refer to a set of practices for adapting applications to different regions, languages, and cultures. You can accomplish this in your applications or websites by considering the differences in language, scripts, and directionality.

Internationalization, sometimes abbreviated as "I18N," refers to the process of designing products that are functionally independent of interfaces or interactions that rely on a specific language or cultural context. This process enables products or services to be equally experienced by people using any language or from any cultural background. Localization, sometimes written as "L10N," concerns adapting or customizing a product to some specific areas. Internationalization and localization are both crucial in making your application prosper in a global market.

You should always consider Internationalization during the design process, especially if there is a vision for reaching our Global market. Understanding and capturing possible needs for localization should begin at the outset of any new product development, even if the short-term goal is focused on only one market.

[TOC Component - Interactive React component]

Changing Locales

[MaterialDesignDescription Component - Interactive React component]

Translation

To localize your application content, you should create a string resource file or object that contains the translations for any display text used in your application. Do not have any text hard-coded in your application's UI. To switch your application to another language, you simply toggle the file/object used to populate the UI to the appropriate translation.

Showing Language Options

[Design System Image]

Language options are typically presented in the user preferences or settings section of an application. In the case where users do not require an account, such as on a public-facing website, the language option is expected in the app bar or in the footer. You should provide cues, such as an icon, to help users locate the language selection.

When showing a list of languages, the language names should be presented in their own language (i.e., use "Español", not "Spanish").

Converting Units, Currencies, and Formats

Units, currency, and formats are usually linked with a particular locale. However, there can be differences in usage even within a single locale. You should provide users with the option to change these values (if appropriate for your application) in their locale settings.

Some commonly influenced formats include:

- Phone number
- Names (e.g., given name vs. surname order)
- Address (e.g., street name first or country name first)
- Date and time
- Numbers (comma vs. period)
- Punctuation

Consider your global audience when crafting forms — you may need to support multiple input formats such as for a phone number.

[Design System Image]

Checking for Layout

[Design System Image]

Different languages or writing systems may require different vertical and horizontal space. To ensure flexibility, avoid using fixed dimensions for UI components such as buttons, labels, and badges. Also, consider setting different default display font sizes for languages that may have smaller glyphs.

> ***NOTE:*** *Brightlayer UI recommends using Noto Sans for all non-western languages. You can read more on our Typography page.*

Bidirectionality / Right-to-Left (RTL) Support

[MaterialDesignDescription Component - Interactive React component]

Some languages, such as Arabic and Hebrew, are read from right to left. Applications supporting these languages will need to mirror the UI so the natural flow is from right to left. For example, navigation drawers should be displayed on the right side of the screen, icons should be to the right of the text in buttons, and any icons with directionality should be mirrored.

[Design System Image]

[Design System Image]

[MaterialDesignDescription Component - Interactive React component]

} />

Cultural Differences

[Design System Image]

Addressing cultural differences is the key concern of localization. Sometimes certain UI elements convey different meaning in different cultures. For example, in America, the red color in the stock market indicates a decline, while in China the same color indicates a rise. A thumbs-up gesture usually expresses an approval in English-speaking country, yet the gesture can have a negative meaning in Iraq. When designing for a new market, you are encouraged to have a local consultant run an audit for your application.

When you are designing for webpages, instead of applications, your primary goal is to advertise your products and services, and appeal to the local market. The language tone, page style, and choice of image assets are expected to be tailored for different targeted culture. Likewise, your user study and market research conclusions derived under a certain cultural background might not apply to another culture.

[MaterialDesignDescription Component - Interactive React component]

} />

[MaterialDesignDescription Component - Interactive React component]

} />

[MaterialDesignDescription Component - Interactive React component]

} />

[MaterialDesignDescription Component - Interactive React component]

} />

Developers

There are a number of available libraries to assist with implementing internationalization in your applications.

Angular

For projects that need the ability to swap between languages, we recommend using ngx-translate. As an alternative, the Angular framework has a powerful built-in i18n tool available, but it requires a separate web app be built for each supported language (e.g. an English version, a Spanish version, etc). This is ideal for larger projects that don't require in-app language switching and don't support more than a few locales.

React

There are many JS libraries available to help internationalize your applications. We recommend using react-i18next to manage your translations. If you are supporting right-

to-left languages, you should also carefully follow the guide from MUI. If you are using JSS styles with MUI v5, you can follow this RTL example to fix issues with certain styles not being flipped correctly.

React Native

As with React for web, we recommend using react-i18next to manage your translations. If you are supporting right-to-left languages, you can make use of the I18nManager from React Native.