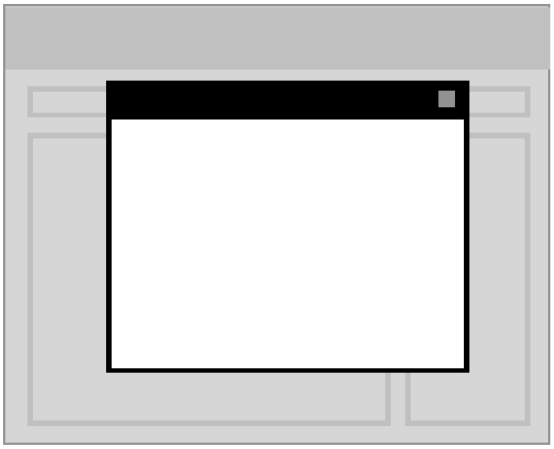
1. Layouts

1.1. Containers & Navigation

1.1.1. Modals

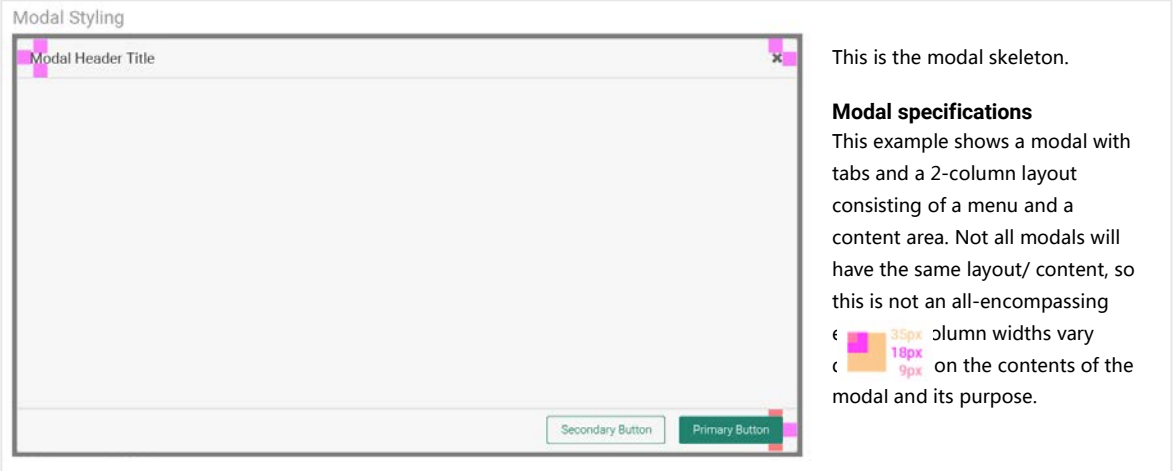
Modals are secondary content areas that allow users focus on a subtask without leaving the current screen. Modals are displayed atop the current screen and may contain varying content. The two types of modals in Sabre interfaces are *functional* modals and *informational* modals.



Design principles

- Display modals in the center of the screen while simultaneously masking the background (such as with a dark overlay).
- Include a clear title that matches text link, icon or button to access the modal.
  - You can ignore this if the content is extremely concise (such as an error message).
- Provide exit points like a "close" icon in the upper right corner or by clicking anywhere outside the modal window. A cancel button may also be used for more complex tasks.
- In complex tasks, consider messages to confirm the users' intents when they click "Close" or somewhere outside the modal to ensure data is not lost unintentionally.
- Do not let modals exceed the height of the viewable space. If the content exceeds this height, separate the content into its own page or simplify it into smaller steps. If this is not possible, display a scroll bar, with a fixed footer for the actions, if applicable.
- Limit the number of buttons appearing in a modal and consider using a single primary button, since the close icon can perform the close/cancel task simultaneously.
- When including more than one button, position them according to the guidelines in the Calls to action section.
- If several modals appear side-by-side or top-and-bottom, make sure they have consistent heights and widths, respectively.
- Determine if your modal is *functional* or *informational.*
  - Functional modals require users to complete some action (such as a transaction) or acknowledge something before proceeding, which disrupts their workflow.

- Informational modals contain data, for instance tabular data that shows results. Users cannot do anything with the information being shown to them. Don't confuse informational modals for popovers or tool tips, when you need to alert users to some information or give them a message.



When to use modals

- To alert or assist users perform a task or to capture secondary information related to form elements on the primary page.
  - Alert users if they are about to perform a task that will delete info or progress.
- When the action does not require its own page to complete.
- When users should focus on a specific action or message.

When not to use modals

- As a popover - background shading should overlay the entire screen.
- For complicated, multi-step tasks, especially on mobile devices. Instead, use the full screen and let users return to the original page via button control.
- Never put a modal within another modal.

- If a secondary action is needed within a modal, use a popover.

Related patterns

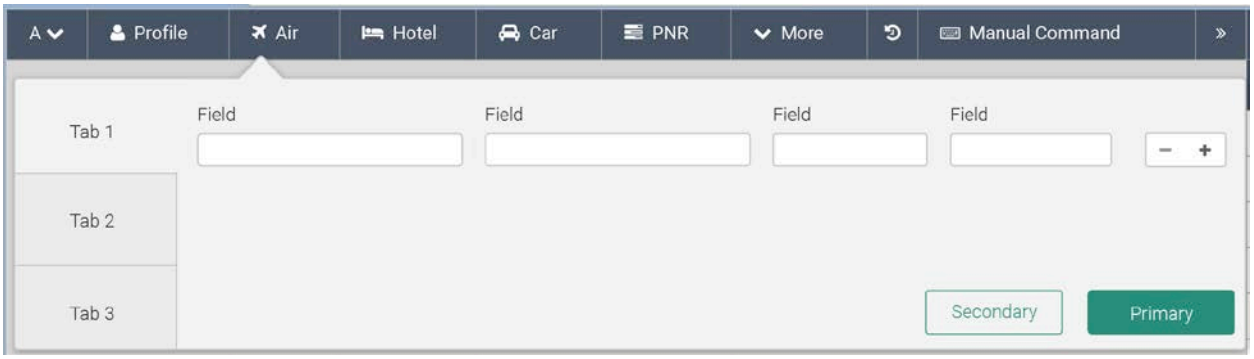
- Popovers

1.1.2. Popovers

Popovers are similar to modals and tooltips in that they are an additional interface that presents some content or actions to the user. Popovers differ from modals in that they do not disable the underlying page content behind them, and they differ from tooltips in that they are actionable and contain more content than mere labels. Popovers allow users to either enter additional data related to an element on a screen or view detailed content related to that element. Popovers are anchored to an element and have a root, whereas a modal is more like a dialog, completely separate from their launch point.

Design principles

- Enable users to activate popovers by a button, link, icon, or form field (opposed to tooltips, which display on hover for non-touch displays—popovers are invoked touch).
- Enable popovers to be closed by clicking a button or clicking outside of the popover.
- Keep popover content relatively short, but scrollable if necessary.
- Position popovers based on the available screen space and layout.



When to use popovers

- To create contextual relationships between a single element displayed on a screen and a more complex set of supporting content or form controls. For example, use a popover when providing additional options related to a filter.
- To provide access to additional detail without requiring the user to change context.
- To display rich content or editable content that is not appropriate for a static tooltip.

When not to use popovers

- For simple content, like additional help text; use a tooltip instead.
- When users must enter in a lot of or complex content; use a modal instead.
- When users must complete some action before proceeding; use a modal instead.

Related patterns

- Modals
- Panels

1.1.3. Accordions

An accordion is a grouped set of collapsible panels that provides access to different sections or other selectable items in a constrained space.

Design principles

- Always set one panel as active/selected by default and at all times.
    - Never display an accordion where all the sections are collapsed.
- Allow each pane of the accordion to be individually expanded (leaving the rest collapsed) by clicking on the panel header.

When to use accordions

- When you can expand only one panel at any given time.

When not to use accordions

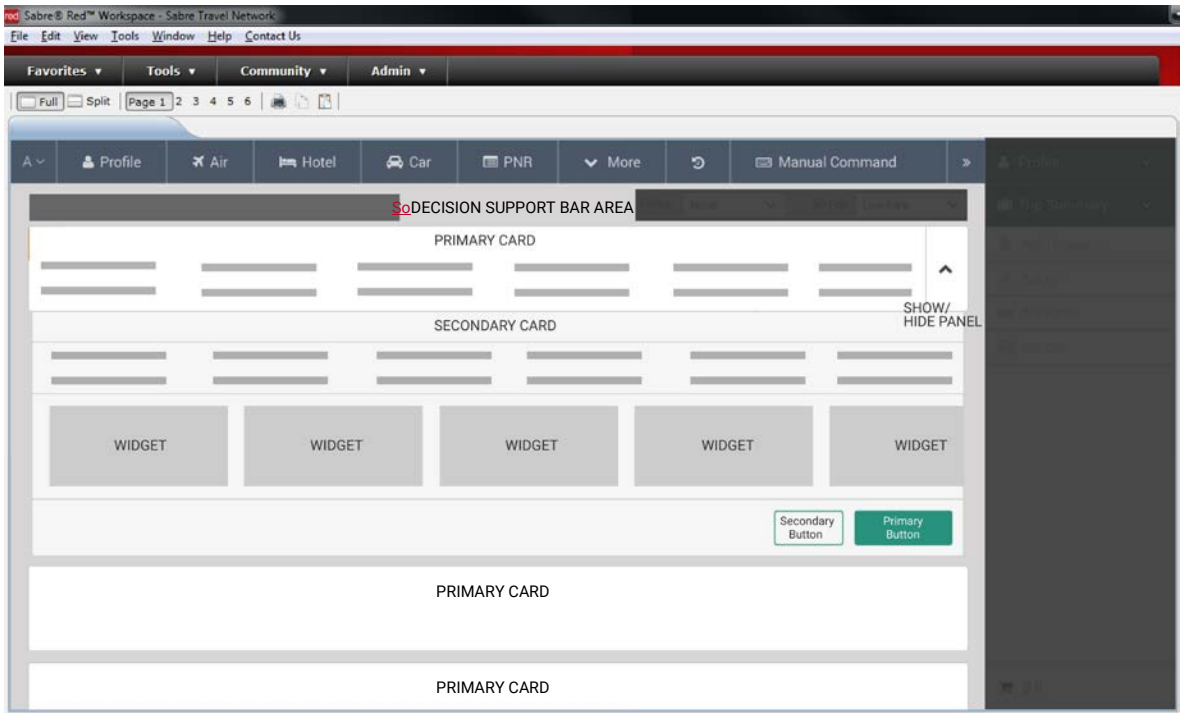
- When multiple panels need to be expanded/collapsed at one time.

Related patterns

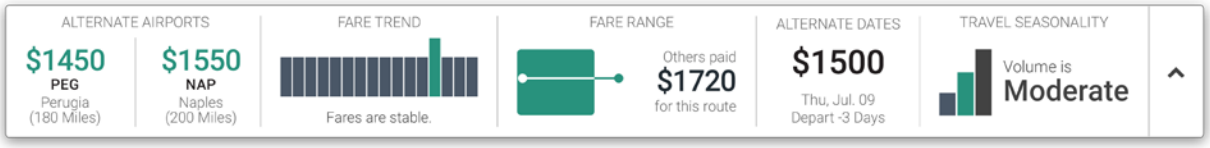
- Modals
- Panels

1.1.4. Cards, Panels, Widgets, and the Decision Support Bar

Cards, Panels, Widgets, and the Decision Support Bar are elements with hierarchical relationships with each other.



Example Decision Support Bar with global widgets



Cards

- Cards refer to containers for the primary results of a search query.
- Cards are the parent containers for panels, which are the parent containers for widgets.
- The primary purpose of the cards is to display enough information that users can scan them to make an informed decision for what they would like to select as a next step (via panels).
- Cards do not contain any calls to action themselves; users must access the secondary level (panels), or perhaps the tertiary level (widgets) to carry out tasks. Note: By using terminal commands, users can perform tasks directly from the cards results.

Panels

- Panels contain additional content related to the cards in a visually pleasing way.

- Panels may contain a variety of content (images, text, tables, forms, etc.) and can be implemented in a variety of layouts to accommodate the needs of different application and screen types.
- Panels may exist as either *side panels* or *bottom panels*.
- Any calls to action available to the user must be accessed via panels or widgets (with the exception of terminal commands—users can execute terminal commands before expanding any panels).

Widgets

- A widget is a modular piece of functionality that shows up in a panel (or in the decision support bar).
- Widgets are always held within a Panel or the Decision Support Bar (explained further).
- Widgets let users take steps in the terminal interface, often referred to as continuation entries.
- Widgets offer a graphical, WYSIWYG method for performing tasks.
- Widgets take the place of terminal commands, for example, the Fare Rules and Graphical Seat Maps widgets.
- Widgets let users go down paths that lead them back to the main task of booking or selling a flight, hotel, or car.
- Some widgets are purely informational, such as the Weather widget.

The Decision Support Bar

- The Decision Support Bar is a *card* that contains *widgets* that are not specific to any particular card; rather, it exists at the PNR or page level.
- The decision support bar applies to the entire results set of a particular query.

Design principles for Cards, Panels, Widgets, and the Decision Support Bar

- Cards
  - Do not put any calls to action on the cards themselves.
  - Use a *progressive disclosure* mechanism to access the secondary level.
  - If there is a command line available, number cards so users can refer to specific cards via the command line.

- Panels
  - Include a heading either inside or outside of the panel (except when using expand/collapse panels, which always display the heading inside of the panel).
  - If several panels appear side-by-side or top-and-bottom, make sure they have consistent heights and widths, respectively.

- Widgets
  - Ensure that widgets always launch an external modal.
  - When widgets are specific to a particular search result, place them on the cards (e.g., a Graphical Seat Map).
  - When widgets are *not* specific to any one particular result, place them on the Decision Support Bar (e.g., Weather).

  - Design widgets to launch into a separate modal.
  - Understand that, to indicate clickability, the user's cursor will change (but there is not a hover state).

- Decision Support Bar
  - Ensure that widgets on the Decision Support Bar relate to entire searches, not results.
  - The widgets on the Decision Support Bar *can* take users away from the main workflow (opposed to the widgets within the panels). For example, if users find a lower price via a widget, that widget can allow them to reshop, disrupting the current workflow and setting them on a new one.

When to use Cards, Panels, Widgets, or the Decision Support Bar

- Cards
  - For information that is specific to a single search result.
  - For lists you want users to be able to scan to then make a further selection.

- Panels
  - To group/arrange content that isn't in another container, such as a table or tabs.
  - To organize small and/or unevenly sized tables for a more harmonious layout.

- Widgets
  - For additional or non-required tasks you want to give users the option to do
  - For additional information, such as weather or fare rules, seat guru.

- The Decision Support Bar
  - Currently, the Decision support bar is only implemented for air shopping response.
  - For general widgets that don't pertain to a specific card or panel.

When not to use Cards, Panels, Widgets, or the Decision Support Bar

- Cards: when the content is broader than the single search result.
- Panels: when it is not necessary to display large tables and tabbed sets inside of a panel.
- Widgets: for required tasks.
- The Decision Support Bar: for content that pertains to only one specific result.

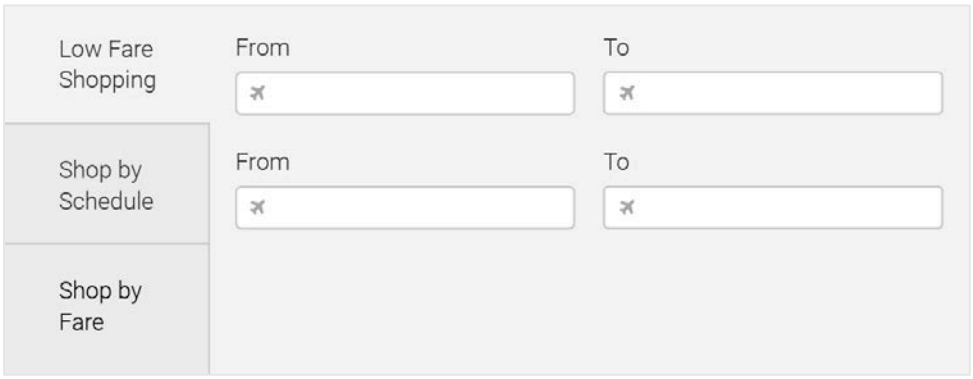
1.2. Navigation

Navigational structures organize sets of related content in separate views. Common visual metaphors for navigation include *tabs* and *pills.* Users can navigate different sections by switching over from one tab/pill to another. Navigation may be either *horizontal* or *vertical.*

Design principles

- Ensure the first navigation item is selected by default to represent users' current location.
- Ensure navigation containers have a fixed minimum and maximum width.

- Never display nested, multiple navigation containers to represent navigation/information hierarchy.
- Organize navigation containers horizontally to span the width of the page/modal or window/section.
- Know when to display *horizontal* or *vertical* navigation depending on your page layout.
- Consider using either *tabs* or *pills* (shown below, respectively) as your nav mechanism.
    - Tabs and pills are functionally equal.



Vertical tabs

Horizontal pills

When to use navigation

- When you have to organize related group of information in a single view.

When not to use navigation

- For grouping of un-related items within a set of navigation tabs.
- For navigating across a linear transactional or a multi-step process, where users have to move in a particular sequence for performing their tasks.
- If all the content is contained within a single view.
- To compare data sets, as users can only view one section at a time.

1.2.1 Pagination

Pagination lets users view subsets of large data that cannot be displayed within a single page, for visual or performance reasons. Pagination can appear either as separated pages (where users can click individual page numbers or relative locations, such as “first” and “last”), or continuous scrolling (such as on Twitter or Facebook), which loads a certain amount of data at a time and only loads more when users have scrolled to a certain threshold.

Design principles

- Separate the data onto separate pages, shown as hyperlinks so users can navigate pages.
- Try to display the pagination navigation at the bottom of the data.
- The numbered page links allow user to navigate across the pagination set.
- Highlight the current page as a visual cue of where users are in the overall data.

- Provide *First* and *Last* links to conveniently allow users to jump to those pages.
- Display the same number of results on each page links e.g. 10 results per page.
- Disable the *Back* link on the first page, as well as the *Next* link on the last page.
- If you have a finite set of results, display all the available number of pages.
- With continuous scrolling, when the system retrieves more data, reset the scroll bar to reflect users' current position after the new data is loaded.

Note: this image is just an example of a pagination mechanism. As with most of the other examples presented in this document, you do not have to follow these exact specifications (e.g. colors, rounded corners, etc.).

When to use pagination

- When there is a large set of data that cannot be viewed all at once.
- When you have a limited vertical screen real estate and would like to display data.

When not to use pagination

- When you have a finite set of data that could be viewed on one page.

2. Form elements

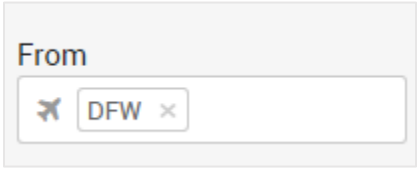
Note: Form elements described below are meant to provide generic design principles and guidelines for forms. This is not prescribing exact visual look and feel and treatment of how they need to be displayed.

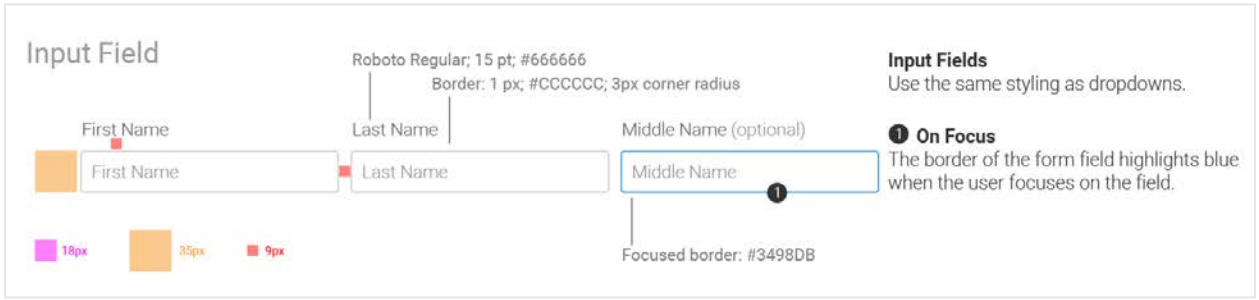
2.1. Text fields

Text fields let users enter or edit text. Text fields respond to interactions with various states, such as *Hover* and *Focused.* Text fields can also have conditions like *Required* or *Disabled.*

Design principles

- Mark any non-required fields with an "Optional" text appendage.
  - Avoid making users enter non-required information as it creates unnecessary burden for users and decreases completion rates (in sign-up/purchase scenarios).
- Use hint text if users' responses have a required format or require more information.
- Enable keyboard selection of auto-complete entries.
  - Select a match by clicking or pressing keyboard arrows.
  - Enter or exit focus by pressing Tab.
  - Complete the form by pressing Enter.
  - Cancel suggestions by pressing Escape.
  - Move the cursor by pressing left/right arrows.
- Mask sensitive information (such as passwords).
  - Allow users to unmask, especially on mobile, where they make more typos.
- Size fields in relation to the data provided by the user.
- Make icons within fields actionable (dropdowns, search, clear, progress indicators, etc.).
- Visually distinguish disabled fields from editable fields.
  - Display disabled fields if more information is required before users can edit the field
- Consider turning users' entries into a pill if it's necessary that they are standardized:





When to use text fields

- When there is not a predefined set of answers from which a user can choose.
- When a user may prefer to paste in a response.
- If users are able to enter more than one value per text field, use a multi-value input.

When not to use text fields

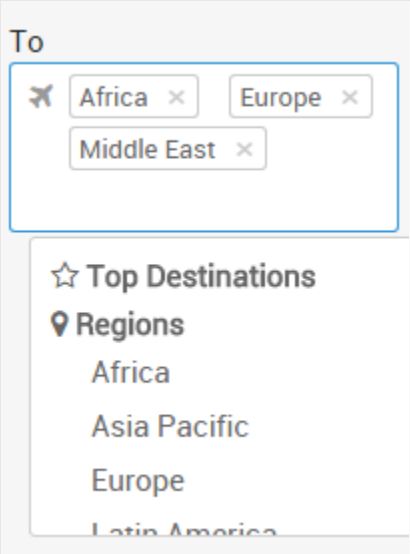
- When options are limited. Instead, consider a select input in these instances to minimize errors caused by incorrect formatting and typing.

Related patterns

- Auto-complete fields
- Search fields

2.1.1. Multi-value text field

Use a multi-value text field when users can enter more than one entire entry per text field.

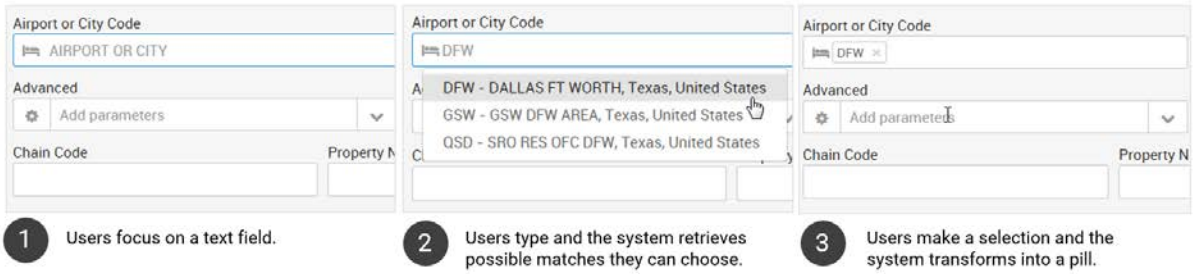


2.2. Auto complete

Auto complete allows users to enter information more quickly by anticipating likely inputs. It may also help them recover from typos and misspellings.

Design principles

- Ensure the user's choice is displayed in the text field once he or she makes a selection.
- Enable keyboard selection of auto-complete entries.
  - Select a match by clicking or pressing keyboard arrows.
  - Enter or exit focus by pressing Tab.
  - Complete the form by pressing Enter.
  - Cancel suggestions by pressing Escape.
  - Move the cursor by pressing left/right arrows.
- Label the text box to match what the user is searching.



When to use auto complete

- For search queries that have many related or similar entries.
- When the user is likely to make a misspelling or typo.
- Use to decrease ambiguity where there is more than one way to enter a query.
- When likely entries are long or complicated to type.

When not to use auto complete

- For private content, such as passwords or credit card/social security numbers.
- When the input required is unknown e.g. Name, Phone numbers.

Related patterns

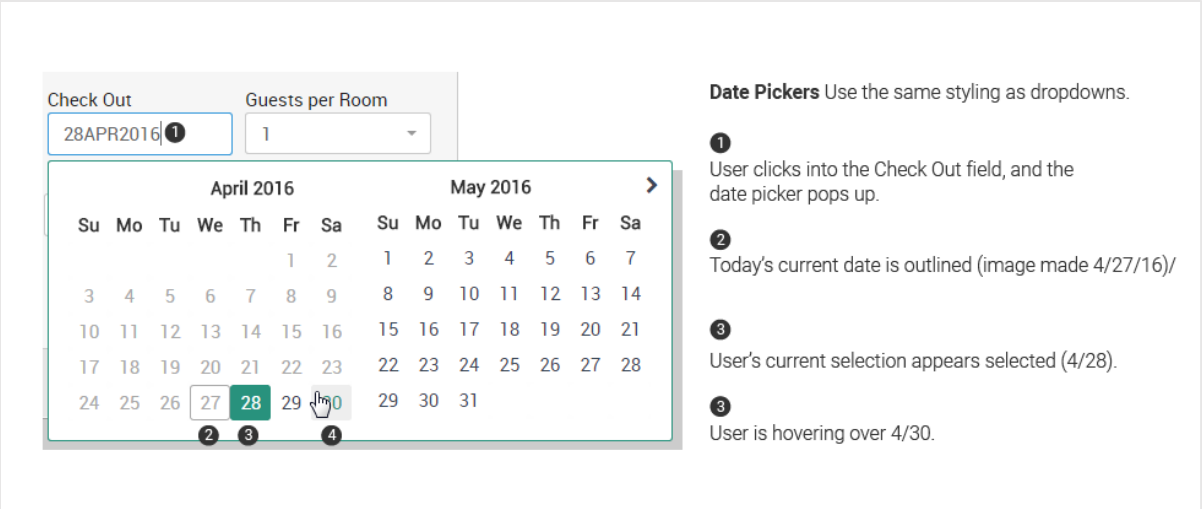
- Text fields
- Search boxes

2.3. Date picker

Date pickers let users select one or more dates by via a virtual calendar. Date pickers can also standardize formatting of date entry (for example day/month/year).

Design principles

- Let users access the date picker as soon as soon as they focus into the text field.
- Ensure users can manually enter the dates as well.
- Make sure date pickers look like real calendars (7 day format, 1 month at a time, etc.).
- By default, display the current month with today's date highlighted.
- Gray out or disable dates that are not available selections to avoid user errors, such as past dates.
- Try to be specific when there are limited calendar months you want to access.



When to use a date picker

- If users must select a range of dates, such as to/from dates.
- When users may want to consider the day of the week in their date selection.
- If users may not know the exact dates they would like to select to give them context of the time of the week, month, or year.

When not to use a date picker

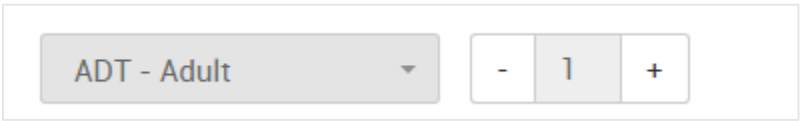
- For dates that are potentially far from today's date (like someone's birthdate).
- For dates that will never change (e.g., a birthday or holiday that is on the same date each year such as Christmas).
- If two or more dates are very spread apart (for example, years apart).

2.4. Number adjustor

A number selector allows the user to add or remove quantities incrementally.

Design principles

- Use two separate controls for decreasing and increasing the value.
- Provide feedback of users' actions by reflecting changes in the text field.



When to use a number adjustor

- When users are most likely to change a value by a small range of increments (for example, when adding passengers to a travel booking).

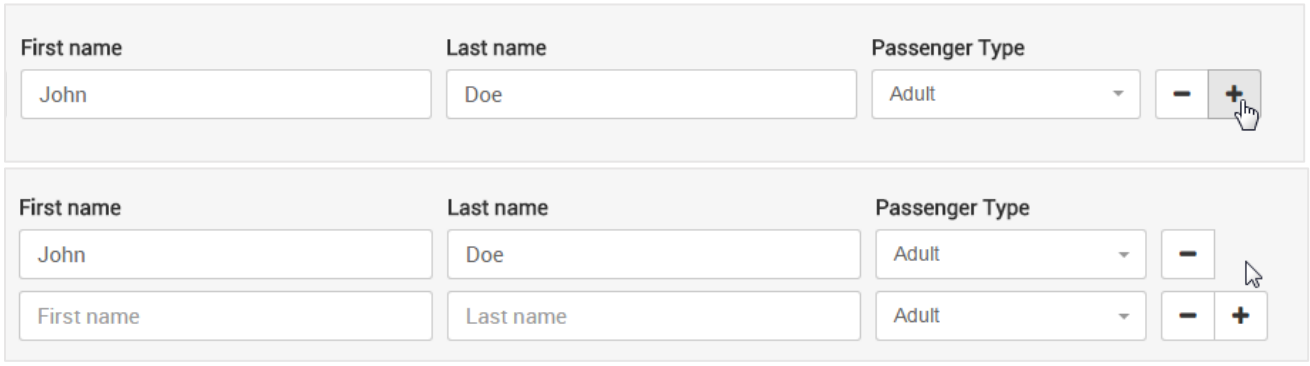
When not to use a number adjustor

- If selecting from a wide range of values, use a text field.

2.5. Add and remove controls

A control to add/remove items lets users change the quantity of any number of UI elements. Use this control when users need to add or remove items of the same formatting.

Design principles

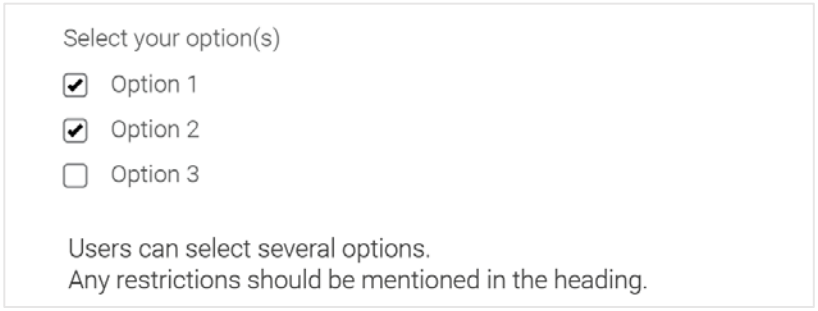


2.6. Checkboxes

Checkboxes allow users to select one or more items from a list or confirm a statement or question (such as acknowledging a privacy statement).

Design principles

- Place the checkbox to the left of the text label.
- Provide a label that describes the checkbox grouping overall.
- Provide clear labels for what each checkbox indicates.
- Use positive grammar (e.g. “Subscribe to this feed,” not “Don’t subscribe to this feed”).
- There is a small set of exceptions to the rule, like “Don’t show this message again.”
- Do not repeat words in the checkbox label from the group label (e.g., if the group label is “At which airports would you like to land?”, don’t make the options “I would like to land at LAX” and “I would like to land at MSY.”)
- Ensure options are related or make sense being in the same group.
- Provide label tags users can click to select/deselect the checkbox.
- Use hover effects for visual cues to users that they can select checkbox labels.
- Provide the ability to select/deselect all the available items.



When to use checkboxes

- When the user may select one or more items from a predetermined list of options.
- When the options are relatively simple or short in length.

When not to use checkboxes

- For a single on/off selection. Use a switch instead.
- When users can only select one option. Use radio buttons instead.

Related patterns

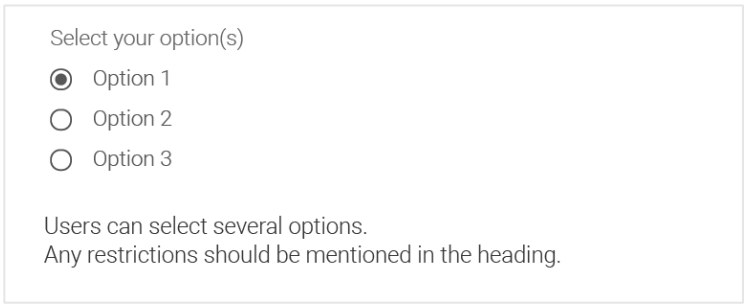
- Radio buttons

2.7. Radio buttons

Radio buttons allow users to make mutually exclusive choices within a set of options. Users can only select one radio button in any given group.

Design principles

- Provide a label that describes the radio button grouping overall.
- Provide clear labels for what each radio button indicates.
- By default, display one radio button as selected.
- Display the set of radio button choices vertically to improve readability.
  - You can use horizontal alignment if you have limited space, such as on mobile.
- Ensure options are related or make sense being in the same group.
- Provide label tags users can click to select/deselect the checkbox.
- Use hover effects for visual cues to users that they can select checkbox labels.



When to use radio buttons

- When users may only select one item from any given list of options.

When not to use radio buttons

- When there is only one option.
- When the options are not related or part of one group.
- When there are only 2 options available; consider using a toggle instead.
- When there are more than 3 options; consider using dropdown box instead.
- For navigation.
- For initiating an action.

Related patterns

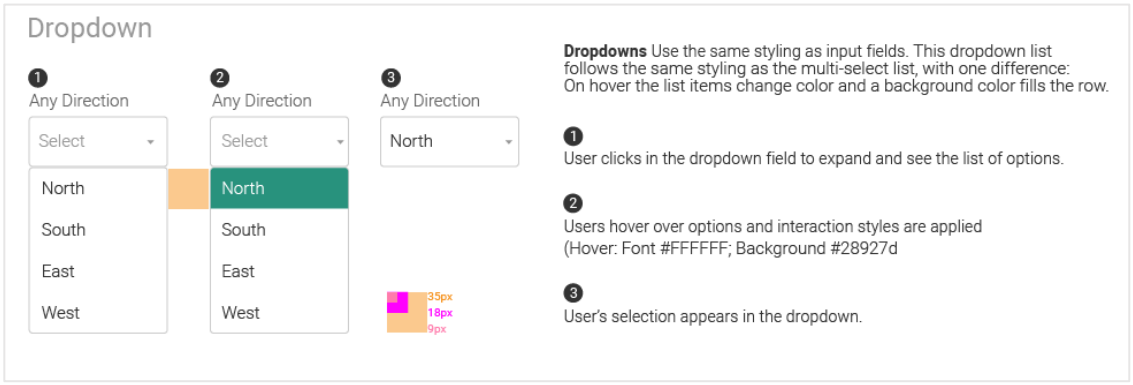
- Checkboxes

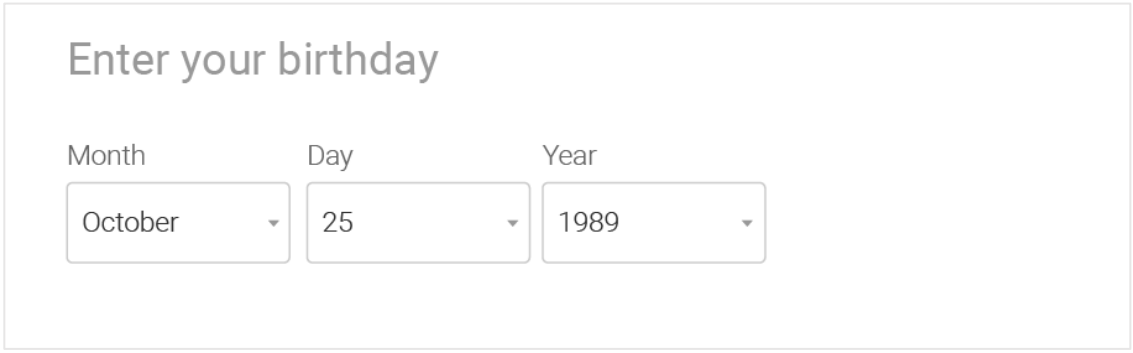
2.8. Dropdown menu

Dropdown menus allow users to select one item from the pre-defined list. A variation of the standard dropdown menu allows users to select more than one item by using an additional text box to input text (with auto complete capability). See multi-select dropdowns (2.8) for more details.

Design principles

- Provide 'Select' as the default value of the dropdown list in its collapsed state.
- When users engage the dropdown, the initial state will be unselected.
  - Once users make a selection, it will be displayed in the collapsed state.
- On hover, option rows will change color to provide feedback.
- Use a dropdown menu group if you have multiple interdependent entries, such as a Month/Day/Year.
  - Make sure items are related enough that they could belong under the same header.
  - Use a horizontal layout if screen real estate permits





When to use dropdowns

- When users can only select one item from a large set of pre-defined data set.
- If there is limited screen real-estate or there are too many options to list as radio buttons.

When not to use dropdowns

- When there is a very limited set of options (2-3). Use radio buttons instead.
- When users can select more than one option. Use checkboxes instead.
- When there are many options where scrolling would be cumbersome. Use multi-select dropdowns instead.

Related patterns

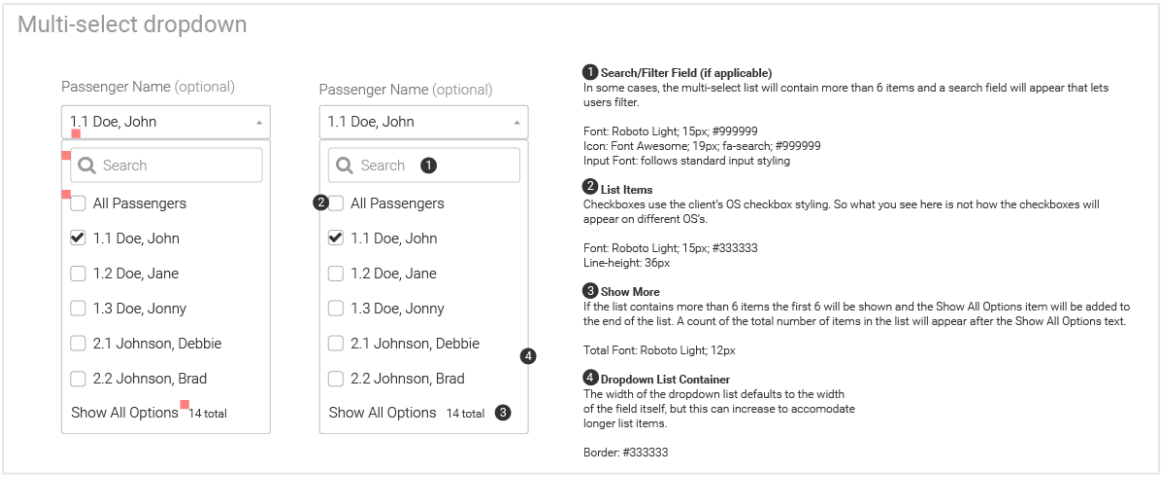
- Multi-select dropdown menus

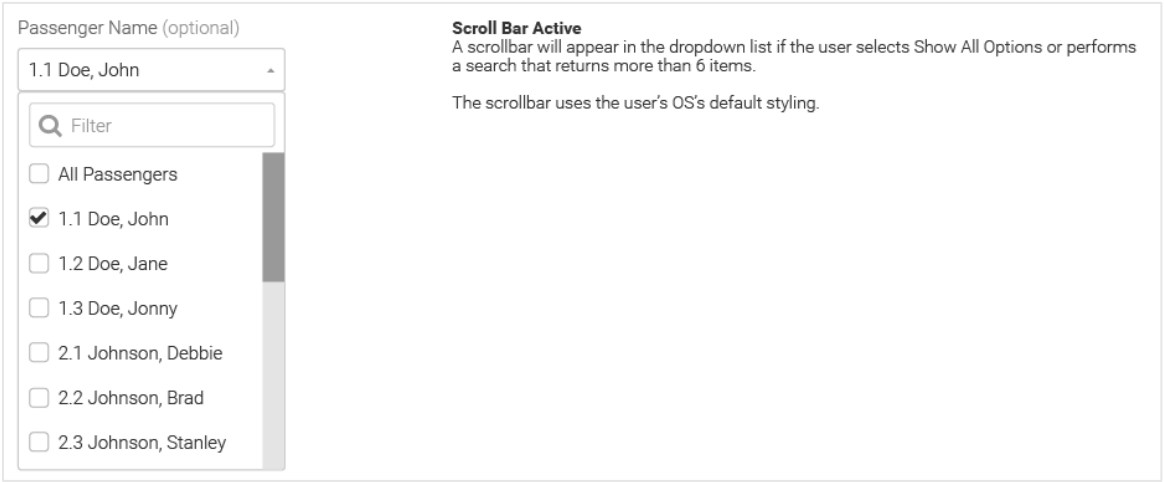
2.9. Multi-select dropdown menu

Multi-select dropdown menus allow users to select multiple items at a time, and support a long list of options while remaining compact.

Design principles

- Display visual feedback in multi-select dropdown menus for all the users' selections.
- Alert users of any constraints, such as a maximum number of selections.
- For mobile, make sure that options have a container of 48 pixels to select the item.





When to use multi-select dropdown menus

- When you have information that could sensibly be formatted as a group of checkboxes, but you have limited screen-real estate.

When not to use multi-select dropdown menus

- If you have enough room to use checkboxes instead.

Related patterns

- Dropdown menus

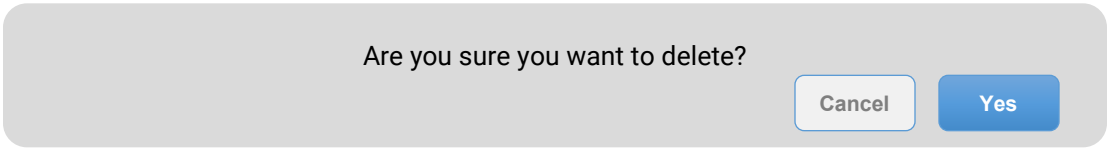
3. Calls to action

3.1. Action buttons

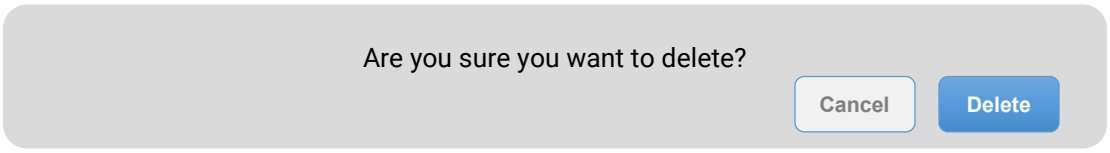
Action buttons are graphical components that allow users to carry out tasks. Action buttons are designed to convey different weights, such as primary and secondary, especially when many action buttons are grouped together.

Design principles

- Button labels should always be clear and concise.
- Button labels should word-for-word match (or as much as possible) the user's actions. For example, the following would be incorrect:



A better example of labeling would be as follows:



- Use task-oriented words that accurately set users' expectations and match their goals.
- Always use consistent button sizes within the same space.
  - If differentiation is needed, use primary, secondary and text button styles.
- Use a loading indicator if a button's action results in a long wait time before completing.
  - This is preferred to a full-screen loading overlay. Note that you will need to prevent other changes being made, such as disabling form inputs or other buttons.
- Know when to use *extra small, small, medium,* and *large* button sizes.
  - Use a smaller button for limited space, such as in a table row or on a mobile screen.
  - Use a medium sized button for most cases.
  - Consider using a larger button on a hero banner or marketing scenario.
- Use an obvious button hierarchy to indicate levels of importance/danger from the design.
- Button labels should always be Section 508 (accessibility) compliant.

Recommended application

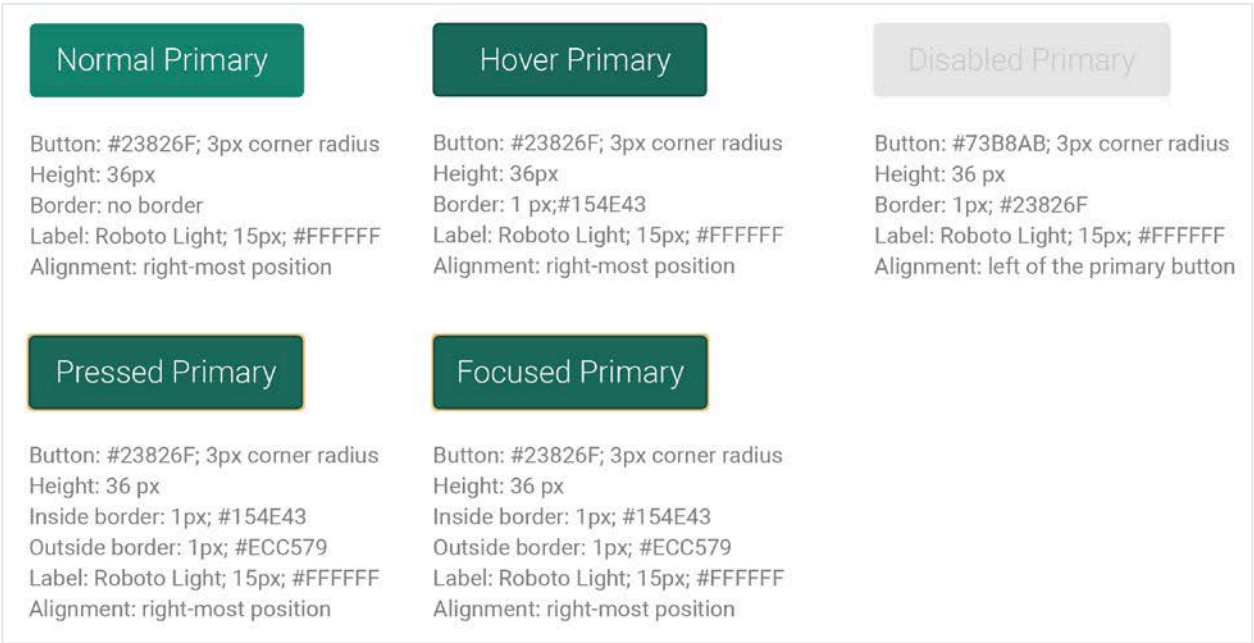
3.1.1. Primary buttons

The primary button allows users to complete main task(s). For example, if the goal is to complete a form, "Submit," "Save," or "Continue" are examples of primary buttons that will likely allow users to make progress on their primary goal.

Design principles

- Limit to a single primary button on a screen or a single button within a panel.
- Primary buttons should appear in the right-most position of a button group.
- On mobile or when buttons are stacked vertically, display primary buttons on the top of the stack.

Recommended application



When to use primary buttons

- To indicate the user's primary workflow.
- To indicate the simplest path the user can take.
- For any action that will help users complete or get closer to completing their goals.
- To indicate the desired or next likely action the user will take.

When not to use primary buttons

- For optional actions.
- When the action is truly secondary in nature but it's the only button in that particular spot (in other words, don't design buttons as primary buttons "just because").

Related patterns

- Secondary buttons
- Tertiary buttons
- Negative/danger buttons
- Icon buttons

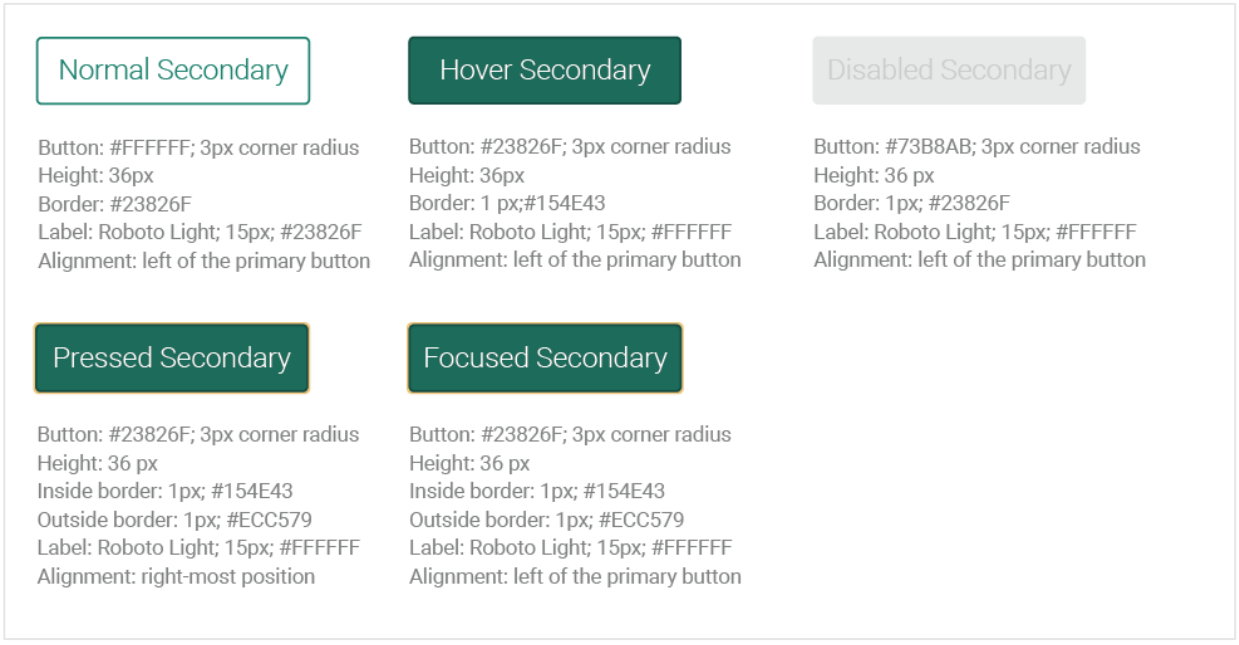
3.1.2. Secondary Buttons

Secondary buttons often allow people to retract data they've entered or complete tasks that are not necessary to the primary workflow. If the goal is to complete a form, "Cancel," "Reset," or "Back," are likely to be secondary to the primary goal of completing a form.

Design principles

- Display secondary actions to the left of the primary action on desktop.
- On mobile, consider displaying secondary buttons to the left or below the primary action.
- Display irreversible functions ("Cancel") on the left.
- Display non-destructive functions ("Save") next to the primary action.
- Reduce the visual prominence of secondary actions to minimize the risk for potential errors and further directs people toward a successful outcome.

Recommended application



When to use secondary buttons

- For non-primary actions.
  - "Cancel" or "Back" are good candidates for secondary buttons.

When not to use secondary buttons

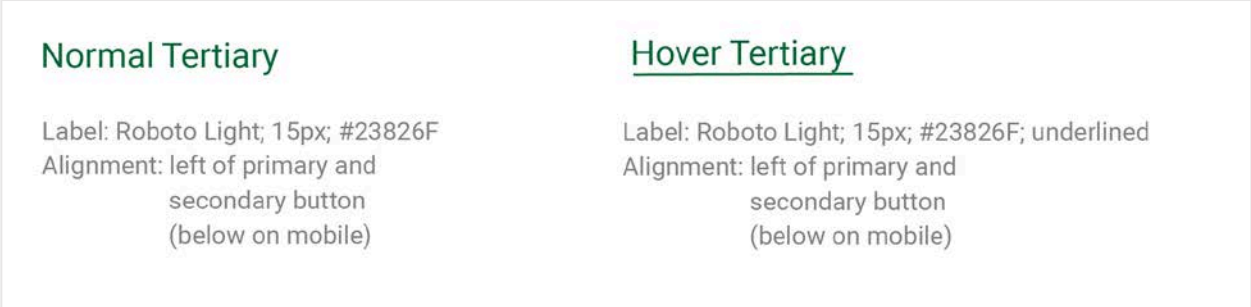
- For steps the user must take to complete their main task(s).

Related patterns

- Primary buttons
- Tertiary buttons
- Negative/danger buttons
- Icon buttons

3.1.3. Tertiary buttons

Tertiary buttons have the least weight out of all button classes, such as "Skip" or "Need Help?"



Design principles

- Deemphasize tertiary buttons to have less weight than primary and secondary buttons.
- Consider styling any tertiary buttons as links.

When to use tertiary buttons

- When the action is too insignificant or infrequent to warrant being a secondary or primary button.
- When the action is not desirable for users to take, but you must still give users the option.

When not to use tertiary buttons

- For steps the user must take to complete their main task(s).
- When a secondary button will suffice.

Related patterns

- Primary buttons
- Secondary buttons
- Negative/danger buttons

- Icon buttons

3.1.4. Warning buttons

Warning buttons convey that the consequences of their actions are likely to be negative to users, such as resulting in an error or deleting data.

Design principles

- Use placement as opposed to color to indicate warning buttons.
  - Place warning buttons away from prominent or typical locations, such as the primary or secondary button locations.
- Use colors relative to the severity of the consequences from highest to lowest.
- Always follow warning buttons with a message that informs users of the consequences of clicking the button.

When to use negative/warning buttons

- For negative or critical actions to convey a clear sense of the action's ramifications.

When not to use negative/warning buttons

- Do not use the negative style just to make a button more prominent.

3.2. Icon buttons

Buttons can also display an icon instead of text.

Design principles

- Include tooltips that describe the action on every icon button.
- Use icon buttons sparingly, and only when there is a clear reason, such as limited space.
- Make sure the icon is clear and self-explanatory, such as an "x" icon for a "Close" action.
- If space allows, consider including supporting text.
  - The icon should always precede any text.

When to use icon buttons

- When screen-real estate is very low, such as on a mobile device or within a table.
- When the icon is relatively well-known or recognizable to users.

When not to use icon buttons

- For actions that are very app-specific or are not easily recognized by users.

Related patterns

- Primary buttons
- Secondary buttons
- Tertiary buttons
- Negative/danger buttons

3.3. Button tool bar

Button tool bars (or sometimes known simply as tool bars) are locations that provide easy access to many settings and features available within the application (especially the most common). Tool bars in certain applications, e.g. Microsoft Word, will change depending on the user's current context and selection.

Design principles

- Ensure the command bar is navigable via keyboard input.
    - Select a match by clicking or pressing keyboard arrows.
    - Enter or exit focus by pressing Tab.
    - Complete the form by pressing Enter.
    - Cancel suggestions by pressing Escape.
    - Move the cursor by pressing left/right arrows.

- Always try to include text with icons, especially for app-specific icons or icons that are not widely-recognized by most users.
- Ensure Section 508 (accessibility) compliance with all command bar text.
- Use a hamburger (☰) or kabob (⋮) icon to indicate that there are hidden actions, if applicable.
- Try to keep command bar labels as short as possible (one word is ideal).
- Keep commands that exist in multiple locations or contexts in the same spot of the command bar.
- Place less-important tools later in the bar's space or within the first few slots of the overflow area. These tools will be visible when the bar has enough screen real estate, but will fall into the overflow area's drop-down menu when there isn't enough room.

When to use a command bar

- When screen-real estate is very low, such as on a mobile device or within a table.
- When the icon is relatively well-known or recognizable to users.

When not to use a command bar

- For actions that are very app-specific or are not easily recognized by users.

4. Tables

4.1. Table display

Tables display large amounts of related data sets based queries to the database. Tables let users view, compare, and analyze information. Tables can be either *informational* (such as for analytical reports) or *transactional*, such as where users can take actions on the attributes available within the data set, e.g. search results.

Design principles

- Ensure tables have the core structural elements of column, rows, and table header.
  - Use additional table elements as necessary, such as a tool bar, sort, filter, or action buttons that manipulate the data presentation or attributes.
- Tables consist of columns, rows, and headers as the core structural components.
  - Tables can include additional elements depending on the structure and size of the table. Some additional elements could be a table toolbar, e.g. sort, filter, and other action buttons to manipulate the data attributes displayed within the table.
- Display the column header as the top row of the table.
- For tables with deeper content, enable vertical scrolling within the fixed table height.
- Limit columns to a manageable number to avoid any horizontal scrolling.
- Alternate row colors (usually gray and white) to increase table legibility and scanability.
- For tables with editable or dynamic content, provide a hover state to help users determine the currently selected row.
- Ensure only one row can be selected or hovered over at any given time unless there are extra controls, such as checkboxes or keyboard enabling of the Control/Command key.
- Design column headings to help users locate and understand the table's data.
  - Use clear, descriptive labels to represent the type of data displayed in the columns.
  - Truncate column headers with ellipses if they are too long or to fit the screen widths.
- Align table content so that it increases scanability and readability.
  - Left align all text, such as names, addresses, emails, phone number, and IDs, et al.
  - Right align the numerical data e.g. Prices, Quantity, Date and Time information.
  - Center align the columns displaying status information e.g. Completed.
- Ensure table titles are clearly labeled.

TOP 5 GROSSING FILMS OF ALL TME

Film Title	Release Date	Distributed By	Worldwide Gross
Avatar	2009	20th Century Fox	$2,787,965,097
Titanic	1997	Paramount	$2,186,772,302
Star Wars: The Force Awakens	2015	Walt Disney Studios	$2,066,484,416
Jurassic World	2015	Universal Pictures	$1,670,400,637
The Avengers	2012	Walt Disney Studios	$1,549,557,910

Example table displaying best practices.

When to use tables

- When you have a lot of data you want to present to the user at one time.

When not to use tables

- When you want to represent trends or relationships among the data.
- When there is only a little data that could rationally be depicted as a bulleted list.

4.2. Table sorting

Table sorting lets users effectively manage and change the order in which data is displayed.

Design principles

- Enable users to sort by clicking on links or buttons in the column headers.
- Let users sort all columns except those containing actions to manipulate the data.
  - If your table has CTAs, group them within an "action column" so users perform actions on the same row with which the actions correspond.
- Let users click on column headers a second time to reverse the sort order. For example, if users click the column header a first time, the rows will be alphabetically sorted in ascending order. If users click the same column header a second time, the rows will reverse their order to be descending from top to bottom.
- Display arrows after the column header to indicate the current sort order.
- Enable a hover state on column headers to indicate the sorting ability.
- Sort textual data (e.g. Name, City) alphabetically and in ascending order by default.
- Sort numerical data (e.g. IDs, Price) numerically and in descending order by default.

When to use sorting

- For large amounts of data so users can change and re-organize content meaningfully.

When not to use sorting

- For limited sets of table rows that don't require any change in order.

4.3. Table filtering

Filters help refine the set of results according to the user's needs. By applying filters, users can reduce the complexity of the data and focus only on the data they need at that time.

Design principles

- Provide filters as categories of options displayed within a dropdown menu.
- When users select a filter, display only the data that corresponds with that filter.
- Let users select multiple filters.
  - As users select filters, update the remaining filters and quantities accordingly, as selecting one filter might reduce the options left in another.

When to use filters

- With a large amount of data that is cumbersome to read and analyze.

When not to use filters

- With a limited set of table rows that doesn't require any change in order.

5. Messages

5.1. Messages

Messages communicate information related to a screen, section of a screen, or field, and may appear by default, in real-time (client-side), or after users submit information (server-side).

Design principles

- Know when to use an intruding prompt vs. a non-intruding prompt.
  - Intruding prompts stop users from continuing and forces them to make a decision or complete an action. A small dialog without the "Close" button displays the message.
  - Non-intruding prompts display while allowing users to continue their tasks.

When to use messages (client-side and server-side)

- Use real-time, or client-side, messages in forms when possible. This prevents the user from making an error and saves the user time. Client-side, real-time messages should always communicate formatting rules. For example: email format.
- Use server-side messages when submitting data is necessary to determine the error state.

When not to use messages

- For system issues that don't impact users' decisions or ability to proceed.

5.1.1. Page-level messages

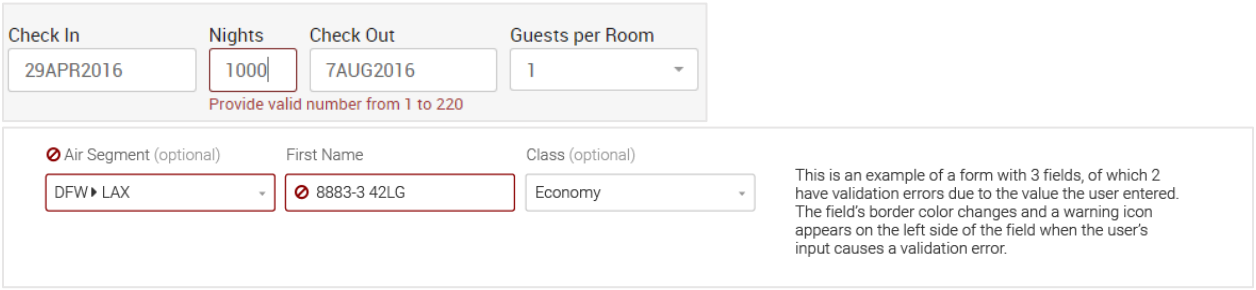
Page-level messages appear at the top of the page and pertain to the overall page content.

5.1.2. Field-level messages (or validation messages)

Field-level messages communicate validation information. See the specific form elements, such as text fields, select, checkboxes, and radio buttons for details about field-level validation.

Design principles

- Always communicate why the input is invalid and how to fix the issue.
- Grammatically refer to users, such as "Password not strong enough. Make sure you include at least one special character."
- If users are missing data, consider giving the field a red fill, border, or other indication.



When to use field-level messages

- When there's a specific format or criterion (e.g. phone number or password strength).
- Present field-level messages as validation feedback once users leave a form field/when the input is directly correlated with levels of success, such as a character count, password strength, required data is missing, and data is invalid.
- Use a character count when there is a character restriction or it is useful to know how many characters have been typed.
    - Ensure it counts down form the max number of characters, if there is one—the count should reflect the remaining character balance.
    - Place the count on the right of a field to prevent any confusion.
    - *Soft character counts* let users keep typing after they've reached the maximum number of characters, and will show a negative balance and turn red.
    - *Hard character counts* do not let the user enter in any additional characters once he or she has reached the maximum number of characters.

When not to use field-level messages

- When inputs will vary in content or length, or if there are no input restrictions.

5.2. Message types

Messages at any level (page-level, field level, or other) are also a particular message type, such as informational, warning, error, and success.

5.2.1. Information messages

Information messages make users aware of something, especially a message of a temporary nature and/or without urgency. For example, the fare price you selected has changed.

When to use information messages

- For information that does not have an explicit positive or negative association.
- Consider displaying in the context of an overall screen or panel, rather than at field level.

5.2.2. Warnings and alerts messages

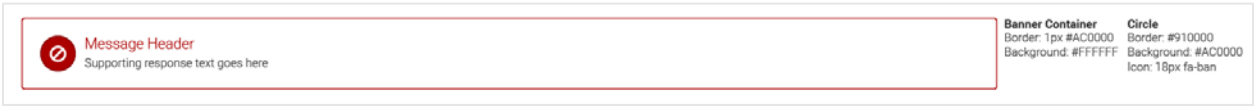
Warning and alert messages allow users to continue their actions, knowing that if the action carries on, a negative result might occur, but it may not be a showstopper. They may also inform users of information, such as a system status, while allowing them to continue their work. Warnings and alerts can exist at any message level, from page-level to field-level.

When to use warnings and alerts messages

- When a user is about to take an action that will cause them to lose work or progress.
- When there is information users should know that is not necessarily negative or urgent.

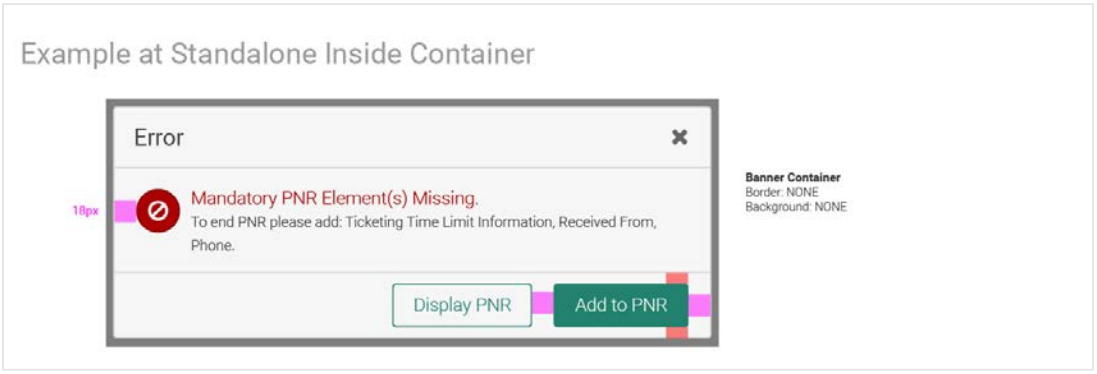
5.2.3. Error messages

Error messages respond to actions users just took; for example, the fare price originally selected is no longer available. See field-level messages on how to give users feedback for invalid input.



Design principles

- Explicitly tell users how to fix the error.
- Use positive language that focuses on advancing users instead of pointing out mistakes.



When to use error messages

- When users must correct something before they can proceed.
    - For real-time messaging, display errors in the context of the field itself.
    - For server-side messaging, display errors at the top of the screen and at the field level. Short forms may omit the top-of-screen message if the required action is apparent to the user.
- Consider using for system errors that prevent users from proceeding down current path.

5.2.4. Success messages

Success messages provide positive feedback that their actions were completed or successful.

When to use success messages

- Use to inform users that they corrected information that was previously causing an error.
- Consider using success messages to confirm that a requested action was completed.
- Use success messages to provide encouragement to users when appropriate.

5.3. Tooltips

Tooltips are brief, descriptive labels displayed when users hover over a target area, such as a button, link, or icon.

Design principles

- Show the tooltip within a very short time window, .25 seconds or less.
- Remove the tooltip promptly when users exit the target area (as quickly as it appears).
- Display the tooltip the entire time the user has the mouse hovering over the target area.
- Set focus to hovered items so users can toggle from mouse to keyboard to navigate.
    - This also allows assistive technologies to detect the focus change and read the text.
- If the line item has an image, place the image in line with the text and
- Support keyboard activation of the target area.
    - Select a match by clicking or pressing keyboard arrows.
    - Enter or exit focus by pressing Tab.
    - Complete the form by pressing Enter.
    - Cancel suggestions by pressing Escape.
    - Move the cursor by pressing left/right arrows.
- For the mouse, use either an arrow or hand pointer, but be consistent throughout.

- Show tooltips either above, below, or to the left or right of the item depending on page layout and other page elements.
- Avoid tooltips for mobile interfaces.

When to use tooltips

- Use to provide brief help text related to a link, button, icon or other element on a screen.
- Use tooltips for desktop applications where users are more likely to be using a mouse.
- When you need to provide more than ALT-attribute text for images.

When not to use tooltips

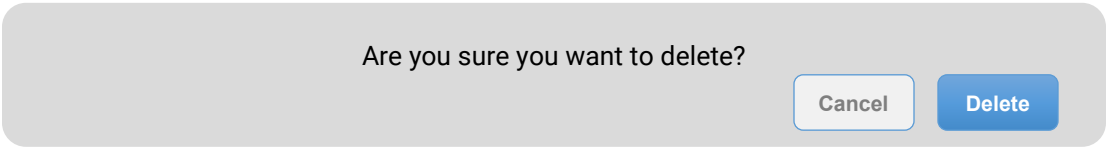
- When you need to display additional content such as images, formatted text and/or form elements. Instead, use a popover.
- For touch or mobile contexts.

5.4. Messages that contain action buttons

Some messages (almost always error or warning messages) may contain action buttons.

Design principles

- Apply the principles in the Calls to Action section for button styling, positioning, and labeling.
- Place buttons in the right (and, if applicable, bottom) corner of the message.



When to use messages with action buttons

- You are alerting users of a consequence and they may want to cancel the operation.

When not to use messages with action buttons

- For notifications or dialogs that would otherwise be closed with an “X” button.

6. Workflow patterns

6.1. Inline expansion (aka linear panels)

Inline expansion flows organize elements and components incrementally, where primary content is displayed upfront and users have to expand to reveal secondary content, like widgets. See Section 4.4. Cards, Panels, Widgets, and the Decision Support Bar for images.

Design principles

- Provide the most important elements in the primary collapsed state so users can understand the information and take actions as appropriate.
- Use chevrons to expand and collapse the secondary layers.
  - Reverse the chevron directions to help users see the different states of the layers. E.g., users must select down chevrons to reveal the secondary layer and vice versa.
- In the secondary layer, include widgets that help users take further action on the content.

When to use

- When there is too much content to fit within a single view, requiring multiple levels.
- Within the multiple levels of elements, display primary content that provides enough information for users to complete their tasks.
- Provide secondary content that is accessible in the context of the primary content.

When not to use

- When you have a finite set of content that be easily organized in a single view.
- When you have a complex group of elements that lead to a transaction flow that increases the depth of the overall container. Use separate page flows or modals instead.

6.2. Hub and spoke

Hub and spoke interaction comprises of a central page (the hub) that contains individual functionalities, CTAs, and components that can launch their own individual task flows (spokes).

For example, an application contains a central hub for managing content. When users click certain components, they are launched to the individual “spokes,” make their changes, and return to the central hub. The updates made in the spoke are reflected within the hub.

Design principles

- Provide a central location (the hub) that contains components and applications that act as links to launch to separate workflows represented as individual pages or modals.
- Allow users to navigate back to the hub after completing tasks on the individual spokes.
- One of the most common examples of the hub and spoke model is on a smartphone. The home is the hub, and all the app icons are the spokes that launch into separate apps.

When to use hub and spoke

- When you want to help users focus on one individual task flow at a time.
- Your application contains several discrete tasks or elements (forms, transactions, entire applications, etc.).

- You don't want to link all the "spokes" to one another for some reason, e.g. space limits, to avoid visual or cognitive clutter, or to enforce the completion of a certain task.
- You want to leave the spokes relatively free to design their own individual navigations and constraints.

When not to use hub and spoke

- When users will want to multi-task on many actions at one time, and will not want to be isolated to one spoke over another at any given time.

6.3 Linear flow

This navigation approach walks users through a multi-step flow, wherein the user has to navigate through a linear, finite number of steps to complete a process.

Design principles

- Place navigation buttons at the bottom of each screen to orient users through the multi-step process.
- Display a progress indicator to keep users informed of their current position in the process. Progress indicators typically display information such as the number of steps completed, users' current location, and the number of steps required to complete.

When to use

- For a multi-step transactional process or registration process.

When not to use

- When you have a simple workflow that can use a single page or modal.
- For non-linear transaction flows where users have to go down different paths to complete their tasks.

7. Other guidelines

7.1. Processing spinner

Processing spinners provide visual feedback to users about the time it takes to complete actions.

Design principles

- Display the spinner until the requested page or modal updates have been processed.
- Consider displaying spinners at the center of the element (page, modal, tab section, etc.).
- Only display spinners if the wait time for a process is more than one second.

7.2. Graphics and images

The following are general guidelines on the use of graphics and images in web applications.

Design principles

- Use graphics and images sparingly and only when required, as page performance and load speed decreases drastically as more or heavier graphics are used.
    - Use graphics formats that load quickly on the end user's browser or platform.
    - Consider using the most preferred file formats for the web, GIF (Graphics Interchange Format) and JPEG (Joint Photographic Experts Group).

7.3. GIFs versus JPEGs

GIFs and JPEGs are two of the most popular graphic file formats. They both have pros and cons for you to consider when deciding which to use.

Design principles

- Use GIFs when there are a lot of the same color(s) and no more than 256 total colors.
- Use GIFs for logos, horizontal/vertical line separators and icons.
- Avoid using GIFs for photographs/graphics that have long stretches of continuous tone in them.
- In GIFs, avoid using gradients and turn off anti-aliasing if possible to minimize file size.
- Use JPEGs for photographs, as they support 16.7 million colors,
- Use JPEGs if you require a relatively small file size that still looks crisp.

7.4. Alt tags

Alt tags refer to the "Alt" attribute within the "IMG" HTML tag. Including an Alt tag in your IMG tag will display a text alternative that appears when users hover over the graphic or image. Alt tags serve as text equivalents of icons and images that do not have descriptive information.

Design principles

- Provide Alt tags for images and icons to indicate their meanings or purposes.
- To enhance the accessibility of your app, always include Alt tags because screen readers cannot read images along, but can read Alt tags aloud.
- Design Alt tags to mirror your tooltip design.