ChatGPT

Chakra UI component overview (v3)

Chakra UI is a React component library that emphasizes accessible, composable and theme-aware UI. The v3 documentation groups components into functional categories such as **layout**, **typography**, **buttons**, **forms**, **collections**, **overlays**, **feedback**, **data display**, **disclosure**, **internationalization** and **utilities**. Each component implements an accessible pattern and exposes props for customization.

Layout components

Chakra's layout primitives help arrange elements and handle spacing and alignment.

Component	Purpose/notes
AbsoluteCenter	Centers a child horizontally/vertically in a relatively positioned parent. The `axis` prop chooses horizontal, vertical or both axes [1].
AspectRatio	Maintains a given ratio (e.g., 16:9) for responsive embeds like videos or images [2]. A `ratio` prop controls the width/height and children are constrained to that ratio.
Bleed	Allows a child to extend ("bleed") outside the padding of its container, useful when an element needs to break its parent's bounds [3].
Box	Low-level component that renders a `<div>` and exposes style props; all other Chakra components build on top of `Box` [4].
Center	Centers its children using flexbox. The parent doesn't need `position: relative`.
Container	Constrains content to a max-width that adapts to breakpoints [5]. Useful for site layouts.
Flex	Implements CSS flexbox; use `direction`, `align`, `justify` and gap props to build flexible layouts [6].
Float	Anchors an element to any edge of its container (e.g., top-right) [6].
Grid & **SimpleGrid**	Provides CSS grid layout. `Grid` offers full control over rows/columns; `SimpleGrid` accepts `columns` or `minChildWidth` for responsive layouts [7].
Group	Groups children so that style props (like hover states) can be applied collectively [8].
ScrollArea	Wraps content in a scrollable area with customizable scrollbars [9].
Separator	Renders a horizontal or vertical line to separate content [10]. Accepts `orientation`, `thickness` and color props.

Component	Purpose/notes
Stack	Stacks children vertically or horizontally with consistent spacing; pass `direction` and `spacing` props to control orientation and gaps [11].
Wrap	Wraps items to the next line when space runs out and adds spacing between them [12].

Typography components

Typography primitives render semantic HTML elements and support style props and variants.

Component	Purpose/notes
Blockquote	Quotes text from an external source. It renders semantic `<blockquote>` and supports a `cite` slot [13].
Code & **Code Block**	Display inline code or multi-line code blocks with syntax highlighting [14]. The code block supports line numbers and copy-to-clipboard.
Em & **Mark**	Emphasize or highlight text. `Em` uses `<em>`, while `Mark` wraps text in `<mark>` and uses a background color [15].
Heading	Renders `<h1>…<h6>` headings with sizes from `xs` to `6xl` [16].
Highlight	Highlights substrings within a longer text; pass a `query` and `styles`.
Kbd	Displays keyboard keys or shortcuts in a monospace style [17].
Link & **LinkOverlay**	Accessible navigation links. `LinkOverlay` stretches a link over a container such as `Card` [15].
List	Creates ordered or unordered lists with customizable spacing and icons [18].
Prose	Styles arbitrary HTML (e.g., blog posts). It applies sensible typography defaults to headings, paragraphs, lists and code.
Text	General purpose text component; supports truncation, decoration, weight and color props [19].

Button components

Component	Purpose/notes
Button	Standard button that triggers actions [20]. Props include `variant` (solid, outline, ghost, etc.), `size`, `colorPalette`, `spinnerPlacement`, `loading`, and `loadingText` [21]. Buttons can contain icons or be square/circular.
IconButton	Button containing only an icon; accessible `aria-label` is required [22].
CloseButton	Icon button used to close modals, dialogs or notifications [23].

Component	Purpose/notes
DownloadTrigger	Button that triggers a file download; pair it with a hidden anchor or programmatic download [24] .
ButtonGroup	Groups several buttons together with shared spacing, orientation and attached styling.

Form components

Chakra's form controls expose root/trigger/content subcomponents for full customization and accessibility. They support `variant`, `colorPalette`, `size`, `isDisabled`, `isInvalid`, default and controlled values.

Component	Purpose/notes
Checkbox	Allows selection of multiple values [25] . Composition uses `Checkbox.Root`, `Checkbox.HiddenInput`, `Checkbox.Control`, `Checkbox.Indicator` and `Checkbox.Label`. Props include `variant` (outline, subtle, solid), `colorPalette`, `size`, `defaultChecked`, `checked` and `indeterminate` [26] . There is also `CheckboxGroup` for grouping options.
Checkbox Card	Variation that wraps a checkbox inside a card; useful for selectable cards [27] .
ColorPicker & **ColorSwatch**	Allow users to pick colors. `ColorPicker` exposes a color area and slider; `ColorSwatch` previews a specific color [28] .
Editable	Inline editing component; starts as text and becomes an input on focus [29] .
Field & **Fieldset**	Provide form control structure: label, help text and error message [30] . `Fieldset` groups controls with a legend.
FileUpload	Allows users to select and upload files from their device [31] .
Input	Standard text input; supports sizes, variants, icons and add-ons.
NumberInput	Input specialized for numbers with increment/decrement steppers [32] .
PasswordInput	Input that hides characters and optionally reveals them via a toggle [33] .
PinInput	Captures short codes or OTPs by rendering multiple boxes [34] .
Radio & **RadioCard**	Allow a single selection from a set. `RadioCard` displays options as cards [35] . Props include `colorPalette`, `size` and `spacing`.
Rating	Displays rating stars or hearts; users can select fractional values [36] .
SegmentedControl	Presents multiple segments where only one can be selected [37] .
Select (Native)	Native `<select>` element styled with Chakra [38] .

Component	Purpose/notes
Slider	Allows selection of a value within a range via a draggable thumb [39]. Supports vertical/horizontal orientation, custom marks and step.
Switch	Binary toggle similar to a checkbox but styled as a switch [40].
Textarea	Multiline text input for longer messages [41].

Collection components

Collection components wrap custom data models (e.g., arrays) and handle keyboard navigation and selection logic.

Component	Purpose/notes
Combobox	Combines a text input with a listbox; allows users to search/filter options and select one or multiple values [42].
Listbox	Stand-alone list of options that can be navigated via keyboard [43]. Often used in conjunction with `Combobox`.
Select (custom)	Custom select widget that renders a hidden native select for form submission [44]. Composition includes `Select.Root`, `Select.Trigger`, `Select.ValueText`, `Select.IndicatorGroup`, `Select.Positioner`, `Select.Content`, `Select.Item` and item groups. It supports sizes, multiple and clearable selections.
TreeView	Displays hierarchical data in a collapsible tree [45]. Nodes can be expanded/collapsed, and icons indicate state.

Overlay components

Overlay components layer content over the page and manage focus trapping and dismissal.

Component	Purpose/notes
ActionBar	Bottom bar that appears when items are selected and contains actions [46]. Composed of `ActionBar.Root`, `ActionBar.Positioner`, `ActionBar.Content`, `ActionBar.SelectionTrigger`, `ActionBar.CloseTrigger` and `ActionBar.Separator`. Controlled via `open` / `onOpenChange` props [47].
Dialog	Displays a modal dialog with backdrop. Offers `Dialog.Root`, `Dialog.Trigger`, `Dialog.Backdrop`, `Dialog.Content`, `Dialog.Header`, `Dialog.Body`, `Dialog.Footer` and `Dialog.CloseTrigger`. Use `open` / `onOpenChange` for control.

Component	Purpose/notes
Drawer	Slides in from the side of the screen and contains header/body/footer [48]. Composition is similar to `Dialog` but anchored to a side. Props include `placement` (left/right/top/bottom), `size`, `open` and `onOpenChange`.
HoverCard	Shows additional content when hovering over an element [49]. Accepts `openDelay`, `closeDelay` and placement props.
Menu	Accessible dropdown menu with nested items [50]. Provides subcomponents for trigger, positioner and content. Items may include checkboxes, radio groups, separators and commands [51].
OverlayManager	Programmatically controls overlay components (open, close, stack).
Popover	Similar to a menu but can contain any content. Contains arrow and can be triggered by click/hover/focus. Supports controlled or uncontrolled states.
ToggleTip	Looks like a tooltip but behaves like a popover; typically triggered by a question mark icon [52].
Tooltip	Displays a small label when the user hovers or focuses on an element [53]. Accepts `label`, `placement`, `openDelay`, `closeDelay` props.

Feedback components

Feedback components convey system states, progress or notifications.

Component	Purpose/notes
Alert	Communicates statuses such as error, warning, success or info [54]. Composition uses `Alert.Root`, `Alert.Indicator`, `Alert.Title`, `Alert.Description` and `Alert.Content`. Props include `status`, `variant` (subtle, solid, outline), `colorPalette` and `size`.
EmptyState	Shows when no data is available, often with an illustration and call-to-action [55].
ProgressCircle & **Progress**	Indicate completion of an operation. `ProgressCircle` draws a circular progress indicator; `Progress` is a horizontal bar [56]. Props include `value`, `max`, `isIndeterminate`, `colorPalette` and `size`.
Skeleton	Placeholder that imitates the shape of content while data is loading [57]. Props control `isLoaded`, `startColor` and `endColor`.
Spinner	Loading spinner that indicates processing [58]. Use `thickness`, `speed`, `emptyColor` and `colorPalette` props.
Status	Displays a small colored indicator (dot or badge) representing statuses such as online/offline [59].

Component	Purpose/notes
Toast	Programmatically triggered transient message overlay [60]. `useToast` hook shows notifications with title, description, status and duration options.

Data-display components

These components present user information, images, lists and tables.

Component	Purpose/notes
Avatar	Shows a user's photo or initials [61]. Composition includes `Avatar.Root`, `Avatar.Image`, `Avatar.Fallback` and `Avatar.Group`. Props allow `size`, `variant`, `shape` and `colorPalette` [62].
Badge	Highlights an item's status or metadata [63]. Props include `variant` (solid, subtle, outline, surface), `colorPalette` and `size` [64].
Card	Presents related content in a container; includes header, body, footer slots and supports elevation and variants [65].
Clipboard	Enables copying text to the clipboard programmatically [66].
DataList	Displays a list of data objects in a structured format [67].
Icon	Renders SVG icons; can import from icon libraries or custom icons [22].
Image	Shows images with fallback, loading and fit options [68].
QR Code	Generates a QR code from a string value [69].
Stat	Displays a statistic with label, number and optional help text [59].
Table	Presents data in rows and columns [70]. Props control size, variant, striped rows and sorting.
Tag	Categorizes or labels content with optional icon and close button [71].
Timeline	Displays events in chronological order [72].

Disclosure components

Disclosure components reveal or hide content and assist navigation.

Component	Purpose/notes
Accordion	Shows/hides multiple sections of related content [73]. Composition uses `Accordion.Root`, `Accordion.Item`, `Accordion.ItemTrigger`, `Accordion.ItemIndicator`, `Accordion.ItemContent` and `Accordion.ItemBody`. Props on `Root` include `multiple` (allow multiple items open), `collapsible`, `orientation`, `variant`, `size`, `colorPalette`, `defaultValue` and `value` for controlled state [74].
Breadcrumb	Indicates the current page's position in a hierarchy [75]. Items are composed of `Breadcrumb.Item`, `Breadcrumb.Link` and `Breadcrumb.Separator`.
Collapsible	Simple collapsible region toggled by a trigger [14]. Accepts `open`, `defaultOpen`, `duration` and easing props.
Pagination	Provides navigational elements for paging through content [76]. Props include `total`, `page`, onPageChange, `size` and `colorPalette`.
Steps	Visualizes progress through multi-step processes [77]. Composed of `Steps.Root`, `Steps.Step`, `Steps.Indicator`, `Steps.Separator` and `Steps.Content`.
Tabs	Displays content in tab panels [78]. Composition includes `Tabs.Root`, `Tabs.List`, `Tabs.Trigger`, `Tabs.Indicator` and `Tabs.Content`. Props control orientation, variant and lazy mounting.

Internationalization components

Component	Purpose/notes
LocaleProvider	Provides locale data (e.g., language and number formats) to components [79].
FormatNumber	Formats numbers according to locale and options such as currency [80].
FormatByte	Formats byte sizes into human-readable strings (KB, MB, etc.) [81].

Utility components

Utility components are helpers for accessibility or advanced composition.

Component	Purpose/notes
Checkmark & **Radiomark**	Visual indicators used inside checkbox and radio controls [82] [83].
ClientOnly	Renders its children only on the client, avoiding server-side rendering mismatches [84].
EnvironmentProvider	Provides environment context when rendering inside iframes, Shadow DOM or Electron [85].

Component	Purpose/notes
For	Loops over an array to render a component for each item; similar to JavaScript `Array.map` but works in JSX syntax [86].
Presence	Animates the entry/exit of components and controls whether they remain mounted [87].
Portal	Renders children into a separate DOM node (e.g., for modals) [88].
Show & **VisuallyHidden**	Conditionally render content (`Show`) and hide content visually while keeping it accessible to screen readers (`VisuallyHidden`) [89] [90].
SkipNav	Provides skip-to-content links for keyboard users [91].
Theme	Allows part of the component tree to be forced to light or dark mode [92].
PortalManager/ OverlayManager	Manage the stacking order of portals and overlays programmatically.

Using these components in prompts

For each component, the code generator should:

1. **Use the component's root element and children properly.** Many Chakra UI components follow a "compound component" pattern. For example, `Accordion` uses `<Accordion.Root>` with nested `<Accordion.Item>`, `<Accordion.ItemTrigger>`, `<Accordion.ItemContent>` and `<Accordion.ItemBody>`. Buttons, drawers, select and other complex components follow similar patterns. If you use the simplified "closed" versions (e.g., `<Button>`), ensure you supply the correct props and children.

2. **Set variant, size and color palette as needed.** Most components accept `variant` (visual style), `size` and `colorPalette` props to quickly adjust appearance [93] [64]. Use the appropriate variant (solid, outline, subtle, ghost, etc.), pick sizes (`xs` to `2xl`) and choose a palette consistent with your design system.

3. **Control state via props or hooks.** Overlay components such as drawers, menus and dialogs accept `open` and `onOpenChange` props for controlled operation [94]. Form inputs accept `value` and `onValueChange` when controlled. Use Chakra's hooks like `useDisclosure` or `useToast` to manage state.

4. **Ensure accessibility.** Chakra UI components implement ARIA roles internally, but you should provide accessible labels (e.g., `aria-label` for `IconButton`, descriptive `title` for alerts), handle focus order and use `VisuallyHidden` for hidden labels when necessary.

5. **Compose with layout primitives.** Arrange components using `Box`, `Flex`, `Grid`, `Stack` and `Wrap`. Use `AspectRatio` for media and `Container` for page widths. Align content with `Center` or `AbsoluteCenter` as appropriate.

By extracting these key details from the Chakra UI v3 documentation, you can craft prompts that instruct the code generator to produce precise, accessible and fully composable Chakra UI components.

---

1 AbsoluteCenter | Chakra UI
https://www.chakra-ui.com/docs/components/absolute-center

2 Aspect Ratio | Chakra UI
https://www.chakra-ui.com/docs/components/aspect-ratio

3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 22 23 24 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 45 49 52 53 55 56 57 58 59 60 65 66 67 68 69 70 71 72 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 Components | Chakra UI
https://www.chakra-ui.com/docs/components/concepts/overview

20 21 93 Button | Chakra UI
https://www.chakra-ui.com/docs/components/button

25 26 Checkbox | Chakra UI
https://www.chakra-ui.com/docs/components/checkbox

44 Select | Chakra UI
https://www.chakra-ui.com/docs/components/select

46 47 Action Bar | Chakra UI
https://www.chakra-ui.com/docs/components/action-bar

48 94 Drawer | Chakra UI
https://www.chakra-ui.com/docs/components/drawer

50 51 Menu | Chakra UI
https://www.chakra-ui.com/docs/components/menu

54 Alert | Chakra UI
https://www.chakra-ui.com/docs/components/alert

61 62 Avatar | Chakra UI
https://www.chakra-ui.com/docs/components/avatar

63 64 Badge | Chakra UI
https://www.chakra-ui.com/docs/components/badge

73 74 Accordion | Chakra UI
https://www.chakra-ui.com/docs/components/accordion