UIKit — API Reference

Pure ES6 component library. No build step, no dependencies. Part of the Evol Soft toolset.

Every component is a plain class: instantiate it, configure it with chainable setter methods, then append `.element` (the underlying DOM node) to the page. Components communicate through `emit` / `on` events rather than parent-child references. Import from a single entry point:

```
import { Button, Grid, Dialog } from './UIKit/index.js';
```

Styling is theme-driven: components read CSS custom properties (`--app-*` , `--bg-*` , `--text-*` , `--border-*`) rather than hardcoded colors, so a new visual theme is a new stylesheet with the same token names — no JS changes required.

Button

Triggers an action. Supports five visual types (`primary` , `danger` , `success` , `warning` , `default`), two sizes, a disabled state, and an optional function-key hint label (e.g. "[F2]") for keyboard-driven workflows.

Input & Textarea

Text entry fields. Both remap the native `input` event to a custom `change` event so listeners fire on every keystroke rather than only on blur — useful for live filtering and validation feedback.

Label

A styled static text element, used for form field captions and section headings within panels and dialogs.

Checkbox & RadioButton

Standard boolean and single-choice inputs, following the native label-wraps-input pattern for accessibility. RadioButton instances sharing a `name` form a mutually exclusive group automatically.

Combo

An autocomplete text input over a searchable list, intended for reference lists up to roughly 200 items, with match highlighting as the user types.

Dropdown

A lightweight custom picker with a trigger button and a popup list, recommended for shorter lists (below the range where `Select` or `Combo` become more appropriate).

Select

Wraps the native `<select>` element. Adopted for large reference lists (50+ items) after a fully custom dropdown ran into positioning problems at that scale.

DatePicker

A custom calendar overlay rather than the native `<input type="date">`, built after the native picker conflicted with custom popup layering and required two clicks to dismiss.

Dialog

Modal window with header/content/footer zones. Maintains an internal stack so multiple nested dialogs always close in the correct order (last opened, first closed).

Panel

A generic container with header/content/footer zones for grouping related content. Re-renders as a whole on update — intended for content that doesn't change frequently.

Tabs

Tab strip backed by a keyed map of panels, supporting dynamic add/remove of tabs at runtime.

List

A scrollable, selectable list of items with single- or multi-select modes and a `select` event.

Grid

The most complex component in the library: sortable/filterable columns, inline cell editing, and column resizing. Internally keeps a separate filtered-rows array so sorting and filtering never mutate the original data set — the source of truth stays untouched.

Table

A plain data table with no sort/filter/pagination — for small, mostly static data where `Grid` would be overkill.

Form

Groups fields with validation and submit handling. `addField()` appends a single field without re-rendering the rest of the form.

DropdownMenu

An "Options ▾" style action menu button: closes on outside click or Escape, supports arrow-key navigation between items.

Link

An anchor-element wrapper that adds a proper `disabled` state (not natively supported on `<a>` tags) via pointer-events and opacity.

VerticalLayout / HorizontalLayout

Three-zone layouts (fixed / stretch / fixed) implemented with absolute positioning rather than flexbox — a pattern carried over from an earlier internal widget library.

Formatters & Validators

Shared helper classes for formatting values for display (numbers, dates, strings) and for validating form input, used internally by `Grid`, `Form`, and other data-bound components.

Base Class

Every component extends a common `Base` class providing `createElement()`, `setProperty()` / `getProperty()`, and the `on()` / `emit()` event pair that all inter-component communication is built on.

CSS Themes & z-index Layers

Four shipped themes (a default Ant-Design-inspired theme plus three alternates) share one token set. Popup layering is standardized project-wide: dialogs, popup menus (Combo/Dropdown/DatePicker/Grid filters), the top-level dropdown menu, and tooltips each occupy a fixed z-index band so overlapping popups never fight each other.
