Flexor Plugin Documentation

Plugin Library

1. **Nested Layouts**
   - What It Does: Enables nested Flexor containers to work seamlessly, adjusting styles to prevent conflicts for complex, hierarchical layouts.
   - How to Use: Add `nested-layouts` to the `data-flexor` attribute of a nested container, e.g., `<div data-flexor="flex col gap-10px nested-layouts">`.
2. **Resize Observer**
   - What It Does: Adjusts child proportions or stacking in real-time based on container size changes using the Resize Observer API.
   - How to Use: Include `resize-observer` in `data-flexor`, e.g., `<div data-flexor="flex row 1 2 1 resize-observer">`.
3. **Virtual Scroll**
   - What It Does: Renders only visible children in large datasets, optimizing performance by minimizing DOM usage during scrolling.
   - How to Use: Add `virtual-scroll` to `data-flexor`, e.g., `<div data-flexor="flex col virtual-scroll">`.
4. **Layout Transition**
   - What It Does: Animates transitions between layout states (e.g., row to column) smoothly using the Web Animation API.
   - How to Use: Include `layout-transition` in `data-flexor`, e.g., `<div data-flexor="flex row gap-10px layout-transition">`.
5. **Equal Heights**
   - What It Does: Ensures all children in a row have the same height, matching the tallest child for uniform layouts.
   - How to Use: Add `equal-heights` to `data-flexor`, e.g., `<div data-flexor="flex row gap-10px equal-heights">`.
6. **Infinite Scroll**
   - What It Does: Appends new children as the user scrolls, fetching content dynamically for endless lists.
   - How to Use: Include `infinite-scroll` in `data-flexor`, e.g., `<div data-flexor="flex col gap-10px infinite-scroll">`.
7. **Accessibility Boost**
   - What It Does: Adds ARIA attributes and keyboard navigation to improve accessibility for screen readers and keyboard users.
   - How to Use: Add `accessibility-boost` to `data-flexor`, e.g., `<div data-flexor="flex row gap-10px accessibility-boost">`.

8. **Content Fit**
   - What It Does: Adjusts the container's size to fit the tallest or widest child, preventing overflow in dynamic layouts.
   - How to Use: Include `content-fit` in `data-flexor`, e.g., `<div data-flexor="flex row gap-10px content-fit">`.
9. **Auto Columns**
   - What It Does: Adjusts the number of columns in a row based on container width, creating a responsive grid with wrapping.
   - How to Use: Add `auto-columns` to `data-flexor`, e.g., `<div data-flexor="flex row gap-10px wrap auto-columns">`.
10. **Drag and Drop**
    - What It Does: Allows users to reorder children by dragging and dropping, with persistent state saved in local storage.
    - How to Use: Include `drag-drop` in `data-flexor`, e.g., `<div data-flexor="flex row gap-10px drag-drop">`.
11. **Lazy Load**
    - What It Does: Delays loading of images or iframes until they're near the viewport, improving initial page load performance.
    - How to Use: Add `lazy-load` to `data-flexor`, e.g., `<div data-flexor="flex col gap-10px lazy-load">`.
12. **Dynamic Spacing**
    - What It Does: Scales gaps between children dynamically based on container size, enhancing responsiveness.
    - How to Use: Include `dynamic-spacing` in `data-flexor`, e.g., `<div data-flexor="flex row dynamic-spacing">`.
13. **Breakpoint Preview**
    - What It Does: Adds a toggle button to preview layout changes at specified breakpoints for debugging.
    - How to Use: Add `breakpoint-preview` to `data-flexor`, e.g., `<div data-flexor="flex row gap-10px stack-500px breakpoint-preview">`.
14. **Aspect Ratio**
    - What It Does: Maintains a fixed aspect ratio (e.g., 16:9) for the container, ideal for media like videos or images.
    - How to Use: Include `aspect-ratio` in `data-flexor`, e.g., `<div data-flexor="flex row aspect-ratio">`.
15. **Conditional Visibility**
    - What It Does: Shows or hides children based on screen size, with smooth transitions for responsive control.
    - How to Use: Add `conditional-visibility` and `data-hide-at` to children, e.g., `<div data-flexor="flex row gap-10px conditional-visibility"><div data-hide-at="500px">`.
16. **Overflow Scroll**
    - What It Does: Adds styled scrolling when children exceed the container's size, managing overflow effectively.
    - How to Use: Include `overflow-scroll` in `data-flexor`, e.g., `<div data-flexor="flex row gap-10px overflow-scroll">`.
17. **Content Alignment**
    - What It Does: Aligns content (e.g., text, images) inside child elements for precise internal positioning.
    - How to Use: Add `content-align` to `data-flexor`, e.g., `<div data-flexor="flex row gap-10px content-align">`.
18. **Sticky**
    - What It Does: Makes the first child stick to the top of the container during scrolling, useful for headers.
    - How to Use: Include `sticky` in `data-flexor`, e.g., `<div data-flexor="flex col gap-10px sticky">`.
19. **Sticky Headers**
    - What It Does: Turns every other child into a sticky header, remaining visible while scrolling through sections.

- How to Use: Add `sticky-headers` to `data-flexor`, e.g., `<div data-flexor="flex col gap-10px sticky-headers">`.

20. **Gap Fill**
    - What It Does: Adds placeholder elements to fill empty spaces in a row or column, maintaining layout symmetry.
    - How to Use: Include `gap-fill` in `data-flexor`, e.g., `<div data-flexor="flex row gap-10px wrap gap-fill">`.
21. **Order Switch**
    - What It Does: Reverses the order of children at a specified breakpoint for responsive reordering.
    - How to Use: Add `order-switch` to `data-flexor`, e.g., `<div data-flexor="flex row gap-10px stack-500px order-switch">`.
22. **Scroll Reveal**
    - What It Does: Fades in children as they scroll into view, enhancing user experience on long pages.
    - How to Use: Include `scroll-reveal` in `data-flexor`, e.g., `<div data-flexor="flex col gap-10px scroll-reveal">`.
23. **Animation**
    - What It Does: Adds entrance animations (e.g., fade-in, slide-up) to children when the page loads.
    - How to Use: Add `animation` to `data-flexor`, e.g., `<div data-flexor="flex row gap-10px animation">`.
24. **Masonry**
    - What It Does: Arranges children in a staggered, masonry-style layout within Flexbox for visual appeal.
    - How to Use: Include `masonry` in `data-flexor`, e.g., `<div data-flexor="flex row wrap gap-10px masonry">`.
25. **Focus Trap**
    - What It Does: Traps keyboard focus within the container, ideal for modals or dialogs.
    - How to Use: Add `focus-trap` to `data-flexor`, e.g., `<div data-flexor="flex col gap-10px focus-trap">`.
26. **Data Binding**
    - What It Does: Binds the layout to a dynamic data source (e.g., JSON), updating children as data changes.
    - How to Use: Include `data-binding` and `data-flexor-data`, e.g., `<div data-flexor="flex row gap-10px data-binding" data-flexor-data='[{"name": "Item 1"}]'>`.
27. **Snap Grid**
    - What It Does: Snaps children to a virtual grid when dragged, enabling precise positioning within the container.
    - How to Use: Add `snap-grid` to `data-flexor`, e.g., `<div data-flexor="flex row snap-grid">`.
28. **Load Balance**
    - What It Does: Distributes child rendering across animation frames for a smoother initial load experience.
    - How to Use: Include `load-balance` in `data-flexor`, e.g., `<div data-flexor="flex row gap-10px load-balance">`.
29. **RTL Support**
    - What It Does: Adjusts layouts for right-to-left languages, reversing rows and aligning text accordingly.
    - How to Use: Add `rtl-support` to `data-flexor`, e.g., `<div data-flexor="flex row gap-10px rtl-support">`.
30. **Theme Switch**
    - What It Does: Toggles between light and dark themes for children with a button, supporting user preferences.
    - How to Use: Include `theme-switch` in `data-flexor`, e.g., `<div data-flexor="flex row gap-10px theme-switch">`.
31. **Error Boundary**
    - What It Does: Wraps children in a try-catch UI, showing fallbacks (e.g., for broken images) if content fails.
    - How to Use: Add `error-boundary` to `data-flexor`, e.g., `<div data-flexor="flex row gap-10px error-boundary">`.
32. **Print Styles**

- What It Does: Optimizes the layout for printing by stacking children and adjusting styles (e.g., no backgrounds).
- How to Use: Include `print-styles` in `data-flexor`, e.g., `<div data-flexor="flex row gap-10px print-styles">`.

33. **Undo Redo**
    - What It Does: Tracks layout changes (e.g., drag-drop) with an undo/redo stack using Ctrl+Z/Y.
    - How to Use: Add `undo-redo` with `drag-drop`, e.g., `<div data-flexor="flex row gap-10px drag-drop undo-redo">`.
34. **Layout Presets**
    - What It Does: Applies predefined layout templates (e.g., hero, sidebar) via a shorthand attribute.
    - How to Use: Include `layout-presets` and `data-preset`, e.g., `<div data-flexor="flex row layout-presets" data-preset="hero">`.
35. **Parallax**
    - What It Does: Adds a parallax effect to child backgrounds, moving them at different speeds on scroll.
    - How to Use: Add `parallax` to `data-flexor`, e.g., `<div data-flexor="flex col gap-10px parallax">`.
36. **Voice Control**
    - What It Does: Allows voice commands (e.g., “stack now”) to adjust the layout using the Web Speech API.
    - How to Use: Include `voice-control` in `data-flexor`, e.g., `<div data-flexor="flex row gap-10px voice-control">`.
37. **Responsive Text**
    - What It Does: Scales child font sizes based on container width for responsive typography.
    - How to Use: Add `responsive-text` to `data-flexor`, e.g., `<div data-flexor="flex row gap-10px responsive-text">.`
38. **Hover Effects**
    - What It Does: Adds scale and color changes to children on hover for interactivity.
    - How to Use: Include `hover-effects` in `data-flexor`, e.g., `<div data-flexor="flex row gap-10px hover-effects">`.
39. **Background Switch**
    - What It Does: Assigns different background colors to children for visual variety.
    - How to Use: Add `background-switch` to `data-flexor`, e.g., `<div data-flexor="flex row gap-10px background-switch">`.
40. **Collaborative Edit**
    - What It Does: Enables real-time collaborative editing of child content via WebSockets.
    - How to Use: Include `collaborative-edit` in `data-flexor`, e.g., `<div data-flexor="flex row gap-10px collaborative-edit">`.
41. **Container Queries**
    - What It Does: Adjusts the layout based on the container’s size (not the viewport) using the Container Queries API, enabling true component-level responsiveness.
    - How to Use: Add `container-queries` to `data-flexor`, e.g., `<div data-flexor="flex row gap-10px container-queries">`.
42. **SSR Prep**
    - What It Does: Prepares Flexor layouts for server-side rendering by serializing initial styles into a `<style>` tag, ensuring compatibility with frameworks like Next.js.
    - How to Use: Include `ssr-prep` in `data-flexor`, e.g., `<div data-flexor="flex row gap-10px ssr-prep">`.
43. **Breakpoint Sync**
    - What It Does: Synchronizes layout breakpoints with CSS custom properties (e.g., --breakpoint-sm), allowing dynamic updates without hardcoding values.
    - How to Use: Add `breakpoint-sync` to `data-flexor`, e.g., `<div data-flexor="flex row gap-10px stack-sm breakpoint-sync">`.
44. **State Manager**
    - What It Does: Persists and restores layout state (e.g., order, visibility) across page loads using URL parameters or local storage for stateful UIs.

- How to Use: Include `state-manager` in `data-flexor`, e.g., `<div data-flexor="flex row gap-10px state-manager">`.

45. **Perf Monitor**
    - What It Does: Tracks and reports layout performance metrics (e.g., render time, reflows) in the console, warning about inefficiencies to aid optimization.
    - How to Use: Add `perf-monitor` to `data-flexor`, e.g., `<div data-flexor="flex row gap-10px perf-monitor">`.
46. **Offline Cache**
    - What It Does: Caches Flexor layouts and assets offline using service workers, enabling PWA functionality with fallback rendering when disconnected.
    - How to Use: Include `offline-cache` in `data-flexor`, e.g., `<div data-flexor="flex row gap-10px offline-cache">`.
47. **AI Layout**
    - What It Does: Optimizes child proportions and stacking automatically based on content analysis (e.g., text length, image size) using a heuristic-based approach.
    - How to Use: Add `ai-layout` to `data-flexor`, e.g., `<div data-flexor="flex row gap-10px ai-layout">`.
48. **Layout Debug**
    - What It Does: Overlays a visual debugging grid or labels showing child proportions, gaps, and breakpoints, toggleable with Ctrl+D for real-time inspection.
    - How to Use: Include `layout-debug` in `data-flexor`, e.g., `<div data-flexor="flex row gap-10px layout-debug">`.
49. **Component Export**
    - What It Does: Exports the current Flexor layout as a reusable HTML/CSS/JS snippet, copyable to the clipboard or downloadable for sharing.
    - How to Use: Add `component-export` to `data-flexor`, e.g., `<div data-flexor="flex row gap-10px component-export">`.
50. **Motion Path**
    - What It Does: Animates children along a custom motion path (e.g., curve, circle) using the Web Animation API for advanced motion design.
    - How to Use: Include `motion-path` in `data-flexor`, e.g., `<div data-flexor="flex row gap-10px motion-path">`.