Framer X Essentials

Unlock the full potential of Framer X with this cheat sheet covering essential shortcuts, core concepts, design tools, and code component basics for rapid prototyping and design.

Core Concepts & Navigation

Keyboard Shortcuts (Mac)

Cmd + N	New Document
Cmd + O	Open Document
Cmd + S	Save Document
Cmd + Shift + S	Save As...
Cmd + W	Close Window
Cmd + Q	Quit Framer X
Space + Drag	Pan Canvas
Cmd + (+/-)	Zoom In/Out
Cmd + 0	Zoom to Fit Canvas
Cmd + 1	Zoom to Actual Size (100%)

Basic Tools

V	Select Tool
F	Frame Tool (Artboard)
R	Rectangle Tool
O	Oval Tool
L	Line Tool
T	Text Tool
P	Pen Tool
C	Code Tool (Create Code Component)
Cmd + D	Duplicate Layer
Cmd + G	Group Layers

Layer Operations Shortcuts

Cmd + [	Send Backward
Cmd +]	Bring Forward
Cmd + Shift + [	Send to Back
Cmd + Shift +]	Bring to Front
Cmd + Shift + L	Lock/Unlock Layer
Cmd + Shift + H	Hide/Show Layer
Cmd + R	Rename Layer
Shift + Click	Select Multiple Layers
Option + Drag	Duplicate Layer by Dragging
Enter	Edit Layer (Text, Vector, etc.)

Framer X UI Explained

Toolbar: Located at the top, contains tools for creating layers, frames, text, and adding assets/components.

Sidebar (Left): Manages pages, layers, and assets. Hierarchical view of your project elements.

Canvas: The main workspace where you design and arrange your frames and layers.

Properties Panel (Right): Displays and allows editing of properties for the selected layer or frame (style, layout, code overrides, etc.).

Preview Window: Shows an interactive preview of your prototype. Accessible via the Play button in the toolbar.

Code Editor: Integrated environment for building and editing code components. Accessed by selecting a code component and clicking 'Edit Code'.

Component Panel: Shows available design and code components from the store, project, or shared libraries.

Design & Layout Tools

Working with Frames

Creating Frames: Use `F` or the Frame tool in the toolbar. Choose a preset device size or draw a custom one.

Frames vs. Groups: Frames are artboards, defining a screen or section. Groups just organize layers. Frames are essential for navigation.

Scrolling: Select a Frame and set its 'Scroll' property in the right panel (Vertical, Horizontal, Both, None).

Overflow: Use the 'Overflow' property to clip content (`Hidden`) or show it (`Visible` , `Scroll`).

Background: Set frame background color, gradient, or image in the style properties.

Renaming: Double-click the frame name in the canvas or layer list, or select and press `Cmd + R` .

Introducing Stacks

What are Stacks? Powerful layout containers that arrange child layers automatically (similar to Flexbox or Auto Layout).

Creating Stacks: Select multiple layers or a group, right-click, and choose 'Create Stack', or use the button in the Properties panel.

Direction: Set layout direction (Horizontal or Vertical).

Distribution: Controls how items are spaced (`Start` , `Center` , `End` , `Between` , `Around`).

Alignment: Aligns items along the cross-axis (`Start` , `Center` , `End` , `Stretch`).

Padding & Spacing: Set padding around the stack's content and spacing *between* items.

Item Controls: Individual stack items can have their own size set (fixed, relative) and alignment overridden.

Creating Links & Navigation

What are Links? Connections between layers (usually buttons or interactive elements) and Frames to simulate screen transitions.

Creating Links: Select the source layer, click the 'Link' icon in the Properties panel, and drag the arrow to the target Frame.

Target: Specify the destination Frame.

Transition: Choose a transition style (Push, Modal, Flip, Fade, None, etc.).

Direction: Set the direction for directional transitions (Left, Right, Up, Down).

Gesture: Define what triggers the link (Tap/Click, Long Press, etc.).

Back Links: Use the 'Back' transition to automatically navigate to the previous screen in the history.

Code Components & Overrides

Code Component Basics

What are Code Components? React components built in Framer X that combine design and code logic. Appear as design assets in the UI.

Creating: Use the Code tool (`C`) or File > New Code Component. Choose a template (Basic, With Props, With State).

Editing: Double-click the component in the Components panel or select it on the canvas and click 'Edit Code'.

Props: Define configurable properties that appear in the Properties panel (`props` object in component code).

State: Manage internal component state using `useState` or `useReducer` hooks.

Children: Use `props.children` to allow design layers to be nested inside the component.

Imports: Import necessary React features (`useState` , `useEffect`) or other modules/libraries.

Exporting: Components must be exported using `export const ComponentName = ...` .

Using Overrides

What are Overrides? Code snippets written in a special file (`Overrides.tsx`) that modify properties or behaviors of design layers *without* turning them into full code components.

Creating Overrides File: File > New Overrides File or click the 'Code' button in the left sidebar.

Applying Overrides: Select a layer on the canvas, go to the 'Code' section in the Properties panel, and select an override function from the dropdown.

Syntax: Overrides are functions that return a properties object. They typically take the layer's existing `props` as an argument.

```
export function AnimateOnTap(): Override {
  return {
    onTap() {
      return {
        scale: 0.8,
        opacity: 0.5,
        transition: { duration: 0.2 }
      }
    },
    onTapEnd() {
      return {
        scale: 1,
        opacity: 1,
        transition: { duration: 0.2 }
      }
    }
  }
}
```

Common Use Cases: Adding tap gestures, animations, state changes based on interactions, fetching data, etc.

Benefits: Keep design and code separate, easily apply interactive behaviors to static design elements.

Tips for Code Components

Use React hooks (`useState` , `useEffect` , `useRef` , `useMemo`) for managing state and side effects.
Define prop types (`PropTypes`) to ensure data consistency and provide documentation.
Start with simple components and gradually add complexity.
Leverage the built-in code editor with TypeScript support and live preview.
Organize your code components into folders within your project.
Remember that Code Components are reusable building blocks. Design layers can be nested within them.

Workflow & Assets

Working with Assets

What are Assets? Reusable design elements (Colors, Text Styles, Images) and components.
Adding Assets: Click the `+` button in the Assets panel (left sidebar). You can add Colors, Text Styles, and Components.
Using Assets: Apply colors and text styles from the Assets panel in the Properties panel. Drag components onto the canvas.
Syncing Assets: Assets can be synced with other design tools (like Sketch) or shared via Framer Teams.
Updating Assets: Edit an asset in the Assets panel, and all instances used in the project will update.
Importing Images: Drag and drop images onto the canvas or use `Cmd + Shift + I` .
Exporting: Select layers or frames and use `Cmd + E` .

Sharing and Collaboration

Publishing: Click the 'Publish' button in the toolbar to create a shareable web link for your prototype.
Private vs. Public: Choose whether the published link is public or requires an invitation.
Embed: Get code to embed your prototype on a website.
Versioning: Framer X includes built-in versioning, accessible from the File menu.
Commenting: Collaborate with others by leaving comments directly on frames in the published prototype.
Teams: For larger teams, Framer Teams offers shared libraries, project permissions, and enhanced collaboration features.

Performance Tips

Avoid deeply nested layer hierarchies where possible.
Optimize images before importing them.
Use Stacks for layout instead of manual spacing where appropriate.
Be mindful of complex code components, especially those performing heavy calculations or many state updates.
Limit the number of layers on a single frame if performance issues arise, potentially breaking content across multiple frames.