DesignLibre
Interaction & Prototyping System

Complete Specification

Version 1.0

Triggers, Actions, Animations, State Machines,
Variables, Conditions, and Prototype Flows

Powered by UnoCSS Transitions & Animations

January 5, 2026

Contents

1 System Overview

1.1 Purpose

The Interaction & Prototyping System transforms static designs into functional, animated prototypes. Users wire components together using a visual node-based system or direct manipulation, defining what happens when users interact with the design.

1.2 Core Concepts

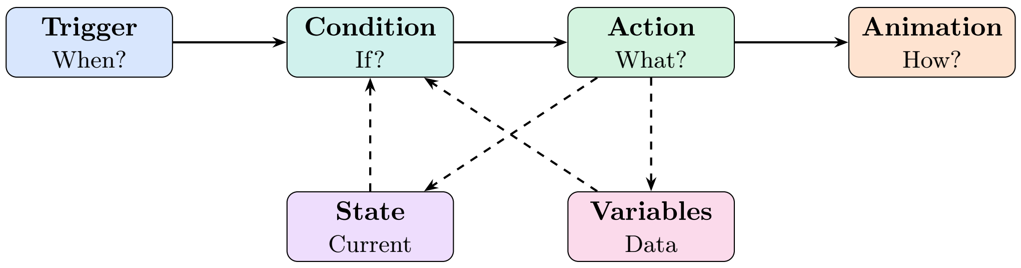

Figure 1: Interaction System Data Flow

- **Trigger**: The event that initiates an interaction (click, hover, scroll, etc.)
- **Condition**: Optional logic that gates whether the action executes
- **Action**: What happens (navigate, show overlay, change state, etc.)
- **Animation**: How the action is visualized (transition type, easing, duration)
- **State**: Component's current variant or mode
- **Variables**: Data that can be read/written and bound to properties

1.3 Interaction Definition Schema

Listing 1: Interaction JSON Schema

```
{
  "id": "interaction_001",
  "name": "Submit Button Click",
  "trigger": {
    "type": "onClick",
    "target": "self"
  },
  "conditions": [
    {
      "variable": "formValid",
      "operator": "equals",
      "value": true
    }
  ],
  "actions": [
    {
      "type": "setVariable",
      "variable": "isLoading",
      "value": true
    },
    {
      "type": "navigate",
      "destination": "frame:success-page",
      "animation": {
        "type": "push",
        "direction": "left",
        "easing": "easeInOut",
        "duration": 400
      }
    }
  ]
}
```

2 Trigger Taxonomy

Triggers

Triggers are events that initiate interactions. They answer: **"When does this happen?"**

2.1 Pointer Triggers

Events from mouse or touch input.

Trigger	**Event**	**Description**
`onClick`	`click`	Mouse click or touch tap completes
`onDoubleClick`	`dblclick`	Two clicks in rapid succession
`onMouseDown`	`pointerdown`	Pointer button pressed (before release)
`onMouseUp`	`pointerup`	Pointer button released
`onHover`	`pointerenter`	Pointer enters element bounds
`onHoverEnd`	`pointerleave`	Pointer exits element bounds
`onLongPress`	Custom	Pointer held for duration (default 500ms)
`onRightClick`	`contextmenu`	Right-click or long-press context

Table 1: Pointer Triggers

2.1.1 Pointer Trigger Properties

Listing 2: Pointer Trigger Configuration

```
{
  "type": "onClick",
  "target": "self",          // "self" | element ID | "parent" | "children"
  "button": "primary",       // "primary" | "secondary" | "middle" | "any"
  "modifiers": {
    "ctrl": false,           // Require Ctrl/Cmd key
    "shift": false,          // Require Shift key
    "alt": false             // Require Alt/Option key
  },
  "preventDefault": true,    // Prevent default browser behavior
  "stopPropagation": false   // Stop event bubbling
}
```

2.2 Drag Triggers

Events for drag-and-drop interactions.

Trigger	Event	Description
`onDragStart`	`dragstart`	Drag operation begins
`onDrag`	`drag`	Element is being dragged (continuous)
`onDragEnd`	`dragend`	Drag operation completes
`onDragEnter`	`dragenter`	Dragged item enters drop zone
`onDragLeave`	`dragleave`	Dragged item leaves drop zone
`onDragOver`	`dragover`	Dragged item over drop zone
`onDrop`	`drop`	Item dropped on drop zone

Table 2: Drag Triggers

2.2.1 Drag Trigger Properties

Listing 3: Drag Trigger Configuration

```
{
  "type": "onDrag",
  "axis": "both",             // "x" | "y" | "both"
  "bounds": "parent",         // "parent" | "viewport" | element ID | null
  "snapToGrid": {
    "enabled": false,
    "size": 8                 // Grid size in pixels
  },
  "dragData": {
    "type": "card",           // Arbitrary data passed to drop zone
    "id": "${self.id}"
  }
}
```

2.3 Gesture Triggers

Touch and multi-touch gestures (mobile/tablet).

Trigger	Gesture	Description
`onSwipe`	Swipe	Quick directional swipe
`onSwipeLeft`	Swipe Left	Swipe in left direction
`onSwipeRight`	Swipe Right	Swipe in right direction
`onSwipeUp`	Swipe Up	Swipe in up direction
`onSwipeDown`	Swipe Down	Swipe in down direction
`onPinch`	Pinch	Two-finger pinch (zoom)
`onPinchIn`	Pinch In	Pinch to zoom out
`onPinchOut`	Pinch Out	Pinch to zoom in
`onRotate`	Rotate	Two-finger rotation
`onPan`	Pan	Single-finger drag/pan

Table 3: Gesture Triggers

2.3.1 Gesture Trigger Properties

Listing 4: Gesture Trigger Configuration

```
{
  "type": "onSwipe",
  "direction": "left",        // "left" | "right" | "up" | "down" | "any"
  "threshold": 50,            // Minimum distance in pixels
  "velocity": 0.3,            // Minimum velocity
  "touches": 1                // Number of touch points required
}
```

2.4 Focus Triggers

Keyboard focus and blur events.

Trigger	Event	Description
`onFocus`	`focusin`	Element receives focus
`onBlur`	`focusout`	Element loses focus
`onFocusVisible`	`focus` + `:focus-visible`	Keyboard focus (not mouse)
`onFocusWithin`	`focusin` (bubbles)	Any child receives focus
`onBlurWithin`	`focusout` (bubbles)	All children lose focus

Table 4: Focus Triggers

2.5 Keyboard Triggers

Key press and release events.

Trigger	Event	Description
`onKeyDown`	`keydown`	Key is pressed
`onKeyUp`	`keyup`	Key is released
`onKeyPress`	`keypress`	Character key pressed (deprecated)
`onShortcut`	Custom	Keyboard shortcut combination

Table 5: Keyboard Triggers

2.5.1 Keyboard Trigger Properties

Listing 5: Keyboard Trigger Configuration

```
{
  "type": "onKeyDown",
  "key": "Enter",             // Key value or code
  "code": "Enter",            // Physical key code
  "modifiers": {
    "ctrl": false,            // Ctrl (Windows) / Cmd (Mac)
    "shift": false,
    "alt": false,
    "meta": false             // Windows key / Cmd key explicitly
  },
  "repeat": false,            // Fire on key repeat
  "global": false             // Listen on document, not element
}

// Shortcut examples
{
  "type": "onShortcut",
  "shortcut": "mod+k",        // mod = Ctrl on Windows, Cmd on Mac
  "global": true
}
{
  "type": "onShortcut",
  "shortcut": "mod+shift+p",
  "global": true
}
```

2.6 Scroll Triggers

Scroll position and intersection events.

Trigger	Event	Description
`onScroll`	`scroll`	Scroll position changes
`onScrollStart`	Custom	Scroll begins (after idle)
`onScrollEnd`	Custom	Scroll ends (debounced)
`onScrollUp`	Custom	Scroll direction is up
`onScrollDown`	Custom	Scroll direction is down
`onViewportEnter`	`IntersectionObserver`	Element enters viewport
`onViewportLeave`	`IntersectionObserver`	Element exits viewport
`onViewportProgress`	`IntersectionObserver`	Visibility ratio changes

Table 6: Scroll Triggers

2.6.1 Scroll Trigger Properties

Listing 6: Scroll Trigger Configuration

```
{
  "type": "onViewportEnter",
  "threshold": 0.5,           // 0-1, portion visible to trigger
  "rootMargin": "0px",        // Margin around viewport
  "once": true                // Only fire once
}

{
  "type": "onScroll",
  "target": "self",           // "self" | "parent" | "window" | element ID
  "throttle": 16,             // Throttle in ms (60fps = 16ms)
  "axis": "y"                 // "x" | "y" | "both"
}

{
  "type": "onScrollProgress",
  "start": "top bottom",      // When element top hits viewport bottom
  "end": "bottom top",        // When element bottom hits viewport top
  "scrub": true               // Link animation to scroll position
}
```

2.7 Form Triggers

Form input and submission events.

Trigger	Event	Description
`onChange`	`change`	Input value changes (on blur)
`onInput`	`input`	Input value changes (immediate)
`onSubmit`	`submit`	Form is submitted
`onReset`	`reset`	Form is reset
`onInvalid`	`invalid`	Input fails validation
`onSelect`	`select`	Text is selected in input

Table 7: Form Triggers

2.8 Media Triggers

Audio/video playback events.

Trigger	Event	Description
`onPlay`	`play`	Media playback starts
`onPause`	`pause`	Media playback pauses
`onEnded`	`ended`	Media playback completes
`onTimeUpdate`	`timeupdate`	Playback position changes
`onVolumeChange`	`volumechange`	Volume changes
`onSeeking`	`seeking`	Seek operation begins
`onSeeked`	`seeked`	Seek operation completes
`onLoadedData`	`loadeddata`	Media data loaded

Table 8: Media Triggers

2.9 Lifecycle Triggers

Component and page lifecycle events.

Trigger	Event	Description
`onLoad`	`load`	Component/page finishes loading
`onMount`	Custom	Component inserted into DOM
`onUnmount`	Custom	Component removed from DOM
`onResize`	`ResizeObserver`	Element size changes
`onOrientationChange`	`orientationchange`	Device orientation changes
`onOnline`	`online`	Network connection restored
`onOffline`	`offline`	Network connection lost
`onVisibilityChange`	`visibilitychange`	Tab visibility changes

Table 9: Lifecycle Triggers

2.10 Time Triggers

Time-based and scheduled events.

Trigger	Mechanism	Description
`afterDelay`	`setTimeout`	Fire after specified duration
`onInterval`	`setInterval`	Fire repeatedly at interval
`onAnimationEnd`	`animationend`	CSS animation completes
`onTransitionEnd`	`transitionend`	CSS transition completes
`onIdle`	`requestIdleCallback`	Browser is idle

Table 10: Time Triggers

2.10.1 Time Trigger Properties

Listing 7: Time Trigger Configuration

```
{
  "type": "afterDelay",
  "delay": 2000,              // Delay in milliseconds
  "repeat": false             // Fire only once
}

{
  "type": "onInterval",
  "interval": 5000,           // Interval in milliseconds
  "limit": 10,                // Maximum iterations (null = infinite)
  "immediate": true           // Fire immediately, then at interval
}
```

2.11 Variable Triggers

React to data changes.

Trigger	Mechanism	Description
`onVariableChange`	Reactive	Variable value changes
`onConditionMet`	Reactive	Condition becomes true
`onStateChange`	Reactive	Component state changes

Table 11: Variable Triggers

2.11.1 Variable Trigger Properties

Listing 8: Variable Trigger Configuration

```
{
  "type": "onVariableChange",
  "variable": "cartItemCount",
  "from": null,               // Previous value (null = any)
  "to": null                  // New value (null = any)
}

{
  "type": "onConditionMet",
  "condition": {
    "variable": "score",
    "operator": "greaterThan",
    "value": 100
  },
  "once": true                // Only trigger once when condition becomes true
}
```

3 Action Types

Actions

Actions define what happens when a trigger fires. They answer: **“What should occur?”**

3.1 Navigation Actions

Actions that change the current view or page.

3.1.1 Navigate

Transition to a different frame, page, or URL.

Listing 9: Navigate Action

```
{
  "type": "navigate",
  "destination": "frame:checkout",     // frame:id | page:id | url:https://...
  "target": "self",                    // "self" | "overlay" | "new-tab"
  "preserveScroll": false,             // Maintain scroll position
  "animation": {
    "type": "push",
    "direction": "left",
    "easing": "easeInOut",
    "duration": 400
  }
}
```

Destination Types:

- `frame:id` — Navigate to frame within current page
- `page:id` — Navigate to different page in prototype
- `url:https://...` — Open external URL
- `back` — Go to previous history entry
- `forward` — Go to next history entry

3.1.2 Open Overlay

Display a component as an overlay above current content.

Listing 10: Open Overlay Action

```
{
  "type": "openOverlay",
  "overlay": "component:modal-confirm",
  "position": {
    "type": "center",                  // "center" | "relative" | "fixed"
    "anchor": null,                    // Element ID for relative positioning
    "offset": { "x": 0, "y": 0 }
  },
  "backdrop": {
    "enabled": true,
    "color": "rgba(0, 0, 0, 0.5)",
    "blur": 4,                         // Backdrop blur in pixels
    "closeOnClick": true
  },
  "animation": {
    "type": "scale",
    "from": 0.95,
    "easing": "easeOut",
    "duration": 200
  }
}
```

Position Types:

- `center` — Centered in viewport
- `relative` — Positioned relative to anchor element (for popovers)
- `fixed` — Fixed position (for drawers, sheets)
- `cursor` — At cursor position (for context menus)

3.1.3 Close Overlay

Dismiss the current or specified overlay.

Listing 11: Close Overlay Action

```
{
  "type": "closeOverlay",
  "target": "current",                 // "current" | "all" | overlay ID
  "animation": {
    "type": "fade",
    "easing": "easeIn",
    "duration": 150
  }
}
```

3.1.4 Scroll To

Scroll to a specific element or position.

Listing 12: Scroll To Action

```
{
  "type": "scrollTo",
  "target": "element:features-section", // element:id | "top" | "bottom" |
      position
  "container": "window",               // "window" | element ID
  "offset": { "x": 0, "y": -80 },      // Offset (e.g., for sticky header)
  "behavior": "smooth"                 // "smooth" | "instant"
}
```

3.2 State Actions

Actions that modify component or application state.

3.2.1 Set State / Swap Variant

Change component to a different state or variant.

Listing 13: Set State Action

```
{
  "type": "setState",
  "target": "self",                    // "self" | element ID
  "state": "pressed",                  // State name
  "animation": {
    "type": "smartAnimate",
    "easing": "easeOut",
    "duration": 200
  }
}
```

```
{
  "type": "swapVariant",
  "target": "self",
  "variant": {
    "size": "large",                  // Variant property
    "color": "primary"                // Can set multiple
  },
  "animation": {
    "type": "smartAnimate",
    "easing": "spring",
    "stiffness": 300,
    "damping": 20
  }
}
```

3.2.2 Toggle State

Toggle between two states.

Listing 14: Toggle State Action

```
{
  "type": "toggleState",
  "target": "self",
  "states": ["collapsed", "expanded"],
  "animation": {
    "type": "smartAnimate",
    "easing": "easeInOut",
    "duration": 300
  }
}
```

3.2.3 Reset State

Return component to default/initial state.

Listing 15: Reset State Action

```
{
  "type": "resetState",
  "target": "self",                   // "self" | element ID | "all"
  "animation": {
    "type": "instant"
  }
}
```

3.3 Variable Actions

Actions that manipulate variable values.

3.3.1 Set Variable

Assign a value to a variable.

Listing 16: Set Variable Action

```
{
  "type": "setVariable",
  "variable": "isLoggedIn",
  "value": true                       // Literal value
}
```

```
{
  "type": "setVariable",
  "variable": "userName",
  "value": "${input.value}"            // Dynamic expression
}

{
  "type": "setVariable",
  "variable": "cartTotal",
  "value": "${cartTotal + item.price}" // Computed expression
}
```

3.3.2 Increment / Decrement

Modify numeric variable.

Listing 17: Increment/Decrement Actions

```
{
  "type": "increment",
  "variable": "counter",
  "amount": 1,                         // Default: 1
  "min": null,                         // Minimum value (optional)
  "max": 100                           // Maximum value (optional)
}

{
  "type": "decrement",
  "variable": "quantity",
  "amount": 1,
  "min": 0
}
```

3.3.3 Toggle Variable

Toggle boolean variable.

Listing 18: Toggle Variable Action

```
{
  "type": "toggleVariable",
  "variable": "darkMode"
}
```

3.3.4 Array Operations

Manipulate array variables.

Listing 19: Array Operation Actions

```
{
  "type": "arrayPush",
  "variable": "todoItems",
  "value": {
    "id": "${generateId()}",
    "text": "${input.value}",
    "completed": false
  }
}

{
  "type": "arrayRemove",
  "variable": "todoItems",
  "index": "${index}"                  // or "where": { "id": "${item.id}" }
}

{
  "type": "arrayUpdate",
  "variable": "todoItems",
  "where": { "id": "${item.id}" },
  "update": { "completed": true }
}
```

3.4 Visual Actions

Actions that affect visual properties without state change.

3.4.1 Animate

Play a defined animation on element.

Listing 20: Animate Action

```
{
  "type": "animate",
  "target": "self",
  "keyframes": [
    { "offset": 0, "transform": "scale(1)" },
    { "offset": 0.5, "transform": "scale(1.2)" },
    { "offset": 1, "transform": "scale(1)" }
  ],
  "options": {
    "duration": 300,
    "easing": "easeInOut",
    "iterations": 1,
    "fill": "forwards"
  }
}

// Or reference a named animation
{
  "type": "animate",
  "target": "self",
  "animation": "pulse",                // Predefined animation name
  "options": {
    "duration": 500
  }
}
```

3.4.2 Set Property

Directly set a visual property (without animation).

Listing 21: Set Property Action

```
{
  "type": "setProperty",
  "target": "self",
  "property": "opacity",
  "value": 0.5
}

{
  "type": "setProperty",
  "target": "element:sidebar",
  "property": "width",
  "value": "300px"
}
```

3.4.3 Show / Hide

Toggle element visibility.

Listing 22: Show/Hide Actions

```
{
  "type": "show",
  "target": "element:tooltip",
  "animation": {
    "type": "fade",
    "easing": "easeOut",
    "duration": 150
  }
}

{
  "type": "hide",
  "target": "element:dropdown",
  "animation": {
    "type": "slideUp",
    "easing": "easeIn",
    "duration": 200
  }
}

{
  "type": "toggleVisibility",
  "target": "element:panel",
  "animation": {
    "type": "fade",
    "duration": 200
  }
}
```

3.5 Data Actions

Actions for data fetching and manipulation.

3.5.1 Fetch Data

Make HTTP request to external API.

Listing 23: Fetch Data Action

```
{
  "type": "fetchData",
  "url": "https://api.example.com/users/${userId}",
  "method": "GET",
  "headers": {
    "Authorization": "Bearer ${authToken}"
  },
  "onSuccess": [
    {
      "type": "setVariable",
      "variable": "userData",
      "value": "${response.data}"
    }
  ],
  "onError": [
    {
      "type": "setVariable",
      "variable": "errorMessage",
      "value": "${error.message}"
    }
  ]
}
```

3.5.2 Submit Form

Submit form data.

Listing 24: Submit Form Action

```
{
  "type": "submitForm",
  "form": "element:contact-form",
  "action": "https://api.example.com/contact",
  "method": "POST",
  "validation": true,                   // Run validation first
  "onSuccess": [
    { "type": "navigate", "destination": "frame:thank-you" }
  ],
  "onError": [
    { "type": "setVariable", "variable": "formError", "value": "${error}" }
  ],
  "onInvalid": [
    { "type": "animate", "target": "self", "animation": "shake" }
  ]
}
```

3.6 Clipboard Actions

Actions for clipboard operations.

Listing 25: Clipboard Actions

```
{
  "type": "copyToClipboard",
  "content": "${codeSnippet}",
  "onSuccess": [
    { "type": "setState", "target": "self", "state": "copied" },
    {
      "type": "afterDelay",
      "delay": 2000,
      "action": { "type": "setState", "target": "self", "state": "default" }
    }
  ]
}
```

3.7 Audio Actions

Actions for sound playback.

Listing 26: Audio Actions

```
{
  "type": "playSound",
  "sound": "asset:click.mp3",           // Asset reference or URL
  "volume": 0.5,                        // 0-1
  "loop": false
}

{
  "type": "stopSound",
  "sound": "asset:background-music.mp3"
}
```

3.8 Utility Actions

Miscellaneous utility actions.

Listing 27: Utility Actions

```
// Log to console (debugging)
{
  "type": "log",
  "message": "Button clicked, value: ${variable}"
}

// Wait/delay
{
  "type": "wait",
  "duration": 500
}

// Execute custom JavaScript
{
  "type": "custom",
  "code": "console.log('Custom code'); return someValue;"
}

// Chain multiple actions in sequence
{
  "type": "sequence",
  "actions": [
    { "type": "setVariable", "variable": "loading", "value": true },
    { "type": "wait", "duration": 1000 },
    { "type": "setVariable", "variable": "loading", "value": false }
  ]
}

// Run actions in parallel
{
  "type": "parallel",
  "actions": [
    { "type": "animate", "target": "element:a", "animation": "fadeIn" },
    { "type": "animate", "target": "element:b", "animation": "fadeIn" }
  ]
}
```

4 Animation System

Animations

Animations define how visual changes occur. They answer: **“How does it look?”**

4.1 Animation Types

4.1.1 Instant

No animation; change happens immediately.

Listing 28: Instant Animation

```
{
  "type": "instant"
}
```

UnoCSS: No transition classes applied.

4.1.2 Dissolve / Fade

Crossfade opacity between states.

Listing 29: Dissolve Animation

```
{
  "type": "dissolve",
  "easing": "easeOut",
  "duration": 200
}
```

UnoCSS: `transition-opacity duration-200 ease-out`

4.1.3 Smart Animate

Automatically interpolate all changed properties between states.

Listing 30: Smart Animate

```
{
  "type": "smartAnimate",
  "easing": "easeInOut",
  "duration": 300,
  "properties": "all"                    // or specific: ["transform", "opacity", "
     background"]
}
```

UnoCSS: `transition-all duration-300 ease-in-out`
Smart animate interpolates:

- Position (x, y) → `transform: translate()`
- Size (width, height)
- Rotation → `transform: rotate()`
- Scale → `transform: scale()`
- Opacity
- Colors (background, border, text)
- Border radius
- Shadows

4.1.4 Move In / Move Out

Slide element in from or out to a direction.

Listing 31: Move In/Out Animation

```
{
  "type": "moveIn",
  "direction": "right",              // "left" | "right" | "top" | "bottom"
  "distance": "100%",                // Distance to travel
  "easing": "easeOut",
  "duration": 300
}

{
  "type": "moveOut",
  "direction": "left",
  "easing": "easeIn",
  "duration": 200
}
```

UnoCSS: `transition-transform duration-300 ease-out` with keyframes:

Listing 32: Move In Keyframes

```
@keyframes moveInRight {
  from { transform: translateX(100%); }
  to { transform: translateX(0); }
}
```

4.1.5 Push

Current content pushes out as new content pushes in.

Listing 33: Push Animation

```
{
  "type": "push",
  "direction": "left",
  "easing": "easeInOut",
  "duration": 400
}
```

4.1.6 Slide Over

New content slides over current (current stays in place).

Listing 34: Slide Over Animation

```
{
  "type": "slideOver",
  "direction": "left",
  "easing": "easeOut",
  "duration": 300,
  "shadow": true                     // Add shadow to sliding element
}
```

4.1.7 Scale

Scale in/out from a point.

Listing 35: Scale Animation

```
{
  "type": "scale",
  "from": 0.95,                        // Starting scale
  "to": 1,                             // Ending scale
  "origin": "center",                  // "center" | "top" | "bottom-left" | etc.
  "easing": "easeOut",
  "duration": 200
}
```

UnoCSS: `transition-transform duration-200 ease-out origin-center`

4.1.8 Flip

3D flip transition.

Listing 36: Flip Animation

```
{
  "type": "flip",
  "axis": "y",                         // "x" | "y"
  "direction": "left",                 // Rotation direction
  "perspective": 1000,                 // Perspective distance
  "easing": "easeInOut",
  "duration": 500
}
```

4.1.9 Morph

Animate between two different shapes/paths.

Listing 37: Morph Animation

```
{
  "type": "morph",
  "easing": "easeInOut",
  "duration": 400
}
```

4.2 Easing Functions

4.2.1 Standard Easings

Easing	CSS	UnoCSS Class
`linear`	`linear`	`ease-linear`
`easeIn`	`ease-in`	`ease-in`
`easeOut`	`ease-out`	`ease-out`
`easeInOut`	`ease-in-out`	`ease-in-out`

Table 12: Standard Easing Functions

4.2.2 Extended Easings

Easing	Cubic Bezier
`easeInSine`	`cubic-bezier(0.12, 0, 0.39, 0)`
`easeOutSine`	`cubic-bezier(0.61, 1, 0.88, 1)`
`easeInOutSine`	`cubic-bezier(0.37, 0, 0.63, 1)`
`easeInQuad`	`cubic-bezier(0.11, 0, 0.5, 0)`
`easeOutQuad`	`cubic-bezier(0.5, 1, 0.89, 1)`
`easeInOutQuad`	`cubic-bezier(0.45, 0, 0.55, 1)`
`easeInCubic`	`cubic-bezier(0.32, 0, 0.67, 0)`
`easeOutCubic`	`cubic-bezier(0.33, 1, 0.68, 1)`
`easeInOutCubic`	`cubic-bezier(0.65, 0, 0.35, 1)`
`easeInQuart`	`cubic-bezier(0.5, 0, 0.75, 0)`
`easeOutQuart`	`cubic-bezier(0.25, 1, 0.5, 1)`
`easeInOutQuart`	`cubic-bezier(0.76, 0, 0.24, 1)`
`easeInQuint`	`cubic-bezier(0.64, 0, 0.78, 0)`
`easeOutQuint`	`cubic-bezier(0.22, 1, 0.36, 1)`
`easeInOutQuint`	`cubic-bezier(0.83, 0, 0.17, 1)`
`easeInExpo`	`cubic-bezier(0.7, 0, 0.84, 0)`
`easeOutExpo`	`cubic-bezier(0.16, 1, 0.3, 1)`
`easeInOutExpo`	`cubic-bezier(0.87, 0, 0.13, 1)`
`easeInCirc`	`cubic-bezier(0.55, 0, 1, 0.45)`
`easeOutCirc`	`cubic-bezier(0, 0.55, 0.45, 1)`
`easeInOutCirc`	`cubic-bezier(0.85, 0, 0.15, 1)`
`easeInBack`	`cubic-bezier(0.36, 0, 0.66, -0.56)`
`easeOutBack`	`cubic-bezier(0.34, 1.56, 0.64, 1)`
`easeInOutBack`	`cubic-bezier(0.68, -0.6, 0.32, 1.6)`

Table 13: Extended Easing Functions

4.2.3 Easing Curve Visualization

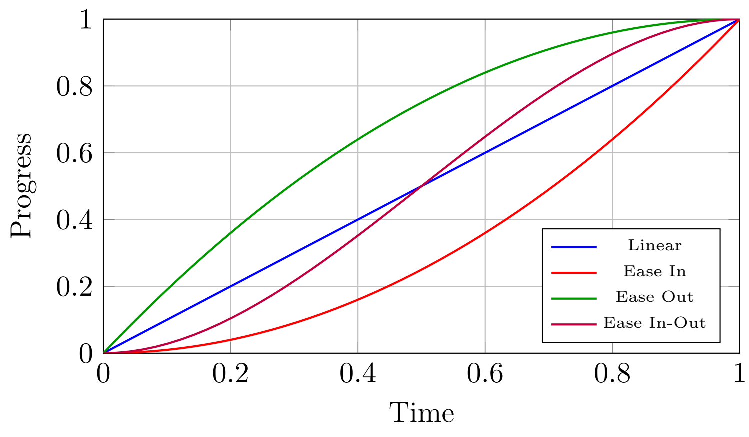

Figure 2: Standard Easing Curves

4.2.4 Spring Physics

For natural, physically-based motion.

Listing 38: Spring Animation

```
{
  "type": "smartAnimate",
  "easing": "spring",
  "stiffness": 300,                    // Higher = faster
  "damping": 20,                       // Higher = less bounce
  "mass": 1,                           // Higher = more inertia
  "velocity": 0                        // Initial velocity
}
```

Spring physics produces natural overshoot and settling behavior. Common presets:

Preset	Stiffness	Damping	Character
`gentle`	120	14	Slow, smooth
`wobbly`	180	12	Bouncy
`stiff`	210	20	Quick, minimal bounce
`slow`	280	60	Slow, no bounce
`molasses`	280	120	Very slow, heavy

Table 14: Spring Presets

4.2.5 Custom Bezier

Define custom easing with control points.

Listing 39: Custom Bezier Easing

```
{
  "easing": {
    "type": "cubicBezier",
    "x1": 0.68,
    "y1": -0.55,
    "x2": 0.27,
    "y2": 1.55
  }
}
```

4.3 Duration

Standard duration scale (in milliseconds):

Token	Value	UnoCSS	Use Case
`instant`	0ms	—	No animation
`fastest`	50ms	`duration-50`	Micro-interactions
`faster`	100ms	`duration-100`	Hover effects
`fast`	150ms	`duration-150`	Buttons, toggles
`normal`	200ms	`duration-200`	Default transitions
`relaxed`	300ms	`duration-300`	Modals, dropdowns
`slow`	500ms	`duration-500`	Page transitions
`slower`	700ms	`duration-700`	Complex animations
`slowest`	1000ms	`duration-1000`	Dramatic reveals

Table 15: Duration Scale

4.4 Delay

Animation delay before starting:

Listing 40: Animation Delay

```
{
  "animation": {
    "type": "fadeIn",
    "delay": 200,                     // Wait 200ms before starting
    "duration": 300
  }
}
```

UnoCSS: `delay-200`

4.5 Stagger

For animating lists/groups, stagger delays each child.

Listing 41: Stagger Animation

```
{
  "type": "stagger",
  "target": "children",
  "animation": {
    "type": "fadeIn",
    "easing": "easeOut",
    "duration": 300
  },
  "stagger": {
    "amount": 50,                     // Delay between each child
    "from": "first",                  // "first" | "last" | "center" | "edges"
    "grid": null                      // For grid layouts: { rows: 3, cols: 4 }
  }
}
```

4.6 Keyframe Animations

Custom multi-step animations.

Listing 42: Keyframe Animation Definition

```
{
  "name": "bounce",
  "keyframes": [
    {
      "offset": 0,
      "transform": "translateY(0)"
    },
    {
      "offset": 0.5,
      "transform": "translateY(-20px)"
    },
    {
      "offset": 1,
      "transform": "translateY(0)"
    }
  ],
  "options": {
    "duration": 500,
    "easing": "easeInOut",
    "iterations": "infinite"          // or number
  }
}
```

UnoCSS Keyframe Definition:

Listing 43: UnoCSS Custom Animation

```
// uno.config.ts
export default defineConfig({
  theme: {
    animation: {
      keyframes: {
        bounce: `{
          0%, 100% { transform: translateY(0); }
          50% { transform: translateY(-20px); }
        }`,
        shake: `{
          0%, 100% { transform: translateX(0); }
          25% { transform: translateX(-5px); }
          75% { transform: translateX(5px); }
        }`,
        pulse: `{
          0%, 100% { opacity: 1; }
          50% { opacity: 0.5; }
        }`,
      },
      durations: {
        bounce: '500ms',
        shake: '300ms',
        pulse: '2s',
      },
      timingFns: {
        bounce: 'ease-in-out',
        shake: 'ease-in-out',
        pulse: 'ease-in-out',
      },
      counts: {
        bounce: 'infinite',
        shake: '1',
        pulse: 'infinite',
      },
    },
  },
})

// Usage: class="animate-bounce" or class="animate-shake"
```

4.7 Predefined Animations Library

Category	Animation	Description
Fade	`fadeIn`	Fade in from transparent
	`fadeOut`	Fade out to transparent
	`fadeInUp`	Fade in while moving up
	`fadeInDown`	Fade in while moving down
Slide	`slideInLeft`	Slide in from left
	`slideInRight`	Slide in from right
	`slideInUp`	Slide in from bottom
	`slideInDown`	Slide in from top
Scale	`scaleIn`	Scale up from 0
	`scaleOut`	Scale down to 0
	`popIn`	Scale with overshoot
	`popOut`	Scale out with overshoot
Attention	`bounce`	Bounce up and down
	`shake`	Shake left and right
	`pulse`	Pulse opacity
	`wiggle`	Rotation wiggle
Flip	`flipX`	3D flip on X axis
	`flipY`	3D flip on Y axis
	`flip`	3D flip diagonal
Rotate	`rotateIn`	Rotate while fading in
	`spin`	Continuous rotation

Table 16: Predefined Animations

5 State Machine Model

States

The state machine tracks component modes and enables variant swapping.

5.1 Component States

Each component can have multiple states, each representing a visual configuration.

5.1.1 State Definition

Listing 44: Component State Definition

```
{
  "component": "Button",
  "states": {
    "default": {
      "background": "bg-blue-500",
      "text": "text-white",
      "shadow": "shadow-md"
    },
    "hover": {
      "background": "bg-blue-600",
      "shadow": "shadow-lg",
      "transform": "-translate-y-0.5"
    },
    "pressed": {
      "background": "bg-blue-700",
      "shadow": "shadow-sm",
      "transform": "translate-y-0"
    },
    "disabled": {
      "background": "bg-gray-300",
      "text": "text-gray-500",
      "opacity": "opacity-50",
      "cursor": "cursor-not-allowed"
    },
    "loading": {
      "background": "bg-blue-500",
      "text": "text-transparent",
      "cursor": "cursor-wait"
      // Spinner overlay shown
    }
  },
  "defaultState": "default",
  "transitions": {
    "default": ["hover", "pressed", "disabled", "loading"],
    "hover": ["default", "pressed"],
    "pressed": ["default", "hover"],
    "disabled": ["default"],
    "loading": ["default"]
  }
}
```

5.1.2 Interactive States (Built-in)

Common states that apply to interactive elements:

State	CSS Pseudo	Description
`default`	—	Initial appearance
`hover`	`:hover`	Pointer over element
`focus`	`:focus`	Element has focus
`focus-visible`	`:focus-visible`	Keyboard focus only
`active`	`:active`	Being pressed/clicked
`disabled`	`:disabled`	Not interactive
`checked`	`:checked`	Checkbox/radio checked
`selected`	`[aria-selected]`	Item is selected
`expanded`	`[aria-expanded]`	Expandable is open
`loading`	Custom	Async operation in progress
`error`	Custom	Validation failed
`success`	Custom	Operation succeeded

Table 17: Interactive States

5.2 Variants

Variants are configurable properties that produce different component versions.

Listing 45: Variant Definition

```
{
  "component": "Button",
  "variants": {
    "size": {
      "options": ["sm", "md", "lg"],
      "default": "md",
      "styles": {
        "sm": { "padding": "px-3 py-1.5", "text": "text-sm" },
        "md": { "padding": "px-4 py-2", "text": "text-base" },
        "lg": { "padding": "px-6 py-3", "text": "text-lg" }
      }
    },
    "variant": {
      "options": ["primary", "secondary", "outline", "ghost"],
      "default": "primary",
      "styles": {
        "primary": { "background": "bg-blue-500", "text": "text-white" },
        "secondary": { "background": "bg-gray-200", "text": "text-gray-800" },
        "outline": { "background": "bg-transparent", "border": "border-2 border-
           blue-500", "text": "text-blue-500" },
        "ghost": { "background": "bg-transparent", "text": "text-blue-500" }
      }
    },
    "fullWidth": {
      "options": [true, false],
      "default": false,
      "styles": {
        "true": { "width": "w-full" },
        "false": { "width": "w-auto" }
      }
    }
  }
}
```

5.3 State Transitions

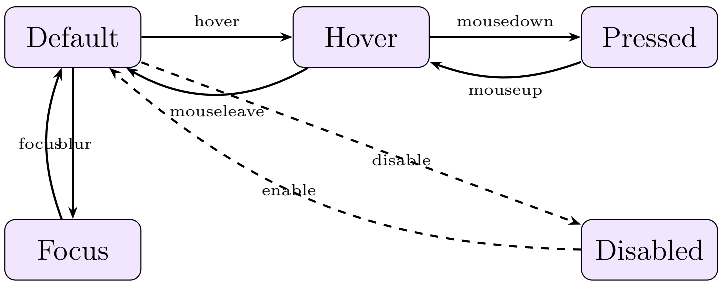

Figure 3: Button State Machine

5.4 Compound States

Multiple states can be active simultaneously:

Listing 46: Compound State Example

```
// Button that is both focused and hovered
{
  "activeStates": ["focus", "hover"],
  "resolvedStyles": {
    // Focus styles take precedence, hover styles merged
    "background": "bg-blue-600",       // from hover
    "ring": "ring-2 ring-blue-500"     // from focus
  }
}
```

UnoCSS: `hover:bg-blue-600 focus:ring-2 focus:ring-blue-500`

5.5 State Persistence

States can be persisted across sessions:

Listing 47: State Persistence

```
{
  "component": "Sidebar",
  "states": {
    "collapsed": { "width": "w-16" },
    "expanded": { "width": "w-64" }
  },
  "persistence": {
    "enabled": true,
    "key": "sidebar-state",
    "storage": "localStorage"          // "localStorage" | "sessionStorage" | "
       none"
  }
}
```

6 Variables System

Variables

Variables store data that can be read, written, and bound to component properties.

6.1 Variable Types

Type	Example Values	Operations
`string`	`"Hello"`, `""`	concat, slice, replace
`number`	`42`, `3.14`, `-10`	add, subtract, multiply, divide
`boolean`	`true`, `false`	and, or, not, toggle
`color`	`"#ff5500"`, `"rgb(255,0,0)"`	lighten, darken, saturate
`object`	`{name: "John"}`	get property, set property
`array`	`[1, 2, 3]`	push, pop, filter, map
`null`	`null`	—

Table 18: Variable Types

6.2 Variable Scopes

Scope	Lifetime	Visibility
`local`	Component instance	Within component only
`page`	Current page/frame	All components on page
`prototype`	Entire prototype session	All pages in prototype
`persistent`	Across sessions	Stored in localStorage

Table 19: Variable Scopes

6.3 Variable Definition

Listing 48: Variable Definitions

```
{
  "variables": [
    {
      "name": "userName",
      "type": "string",
      "scope": "prototype",
      "defaultValue": "Guest",
      "description": "Current user's display name"
    },
    {
      "name": "cartItems",
      "type": "array",
      "scope": "prototype",
      "defaultValue": [],
      "description": "Items in shopping cart"
    },
    {
      "name": "isLoggedIn",
      "type": "boolean",
      "scope": "prototype",
      "defaultValue": false
    },
    {
      "name": "theme",
      "type": "string",
      "scope": "persistent",
      "defaultValue": "light",
      "options": ["light", "dark", "system"]
    },
    {
      "name": "formData",
      "type": "object",
      "scope": "page",
      "defaultValue": {
        "email": "",
        "password": "",
        "rememberMe": false
      }
    }
  ]
}
```

6.4 Variable Binding

Bind variables to component properties for reactive updates.

Listing 49: Variable Binding

```
{
  "component": "WelcomeMessage",
  "bindings": [
    {
      "property": "text",
      "expression": "Hello, ${userName}!"
    },
    {
      "property": "visible",
      "expression": "${isLoggedIn}"
    }
  ]
}

{
  "component": "CartBadge",
  "bindings": [
    {
      "property": "text",
      "expression": "${cartItems.length}"
    },
    {
      "property": "visible",
      "expression": "${cartItems.length > 0}"
    },
    {
      "property": "className",
      "expression": "${cartItems.length > 9 ? 'w-6' : 'w-5'}"
    }
  ]
}
```

6.5 Expressions

Expressions evaluate to values using variables and operators.

6.5.1 Expression Syntax

Listing 50: Expression Examples

```
// Simple variable reference
${userName}

// Object property access
${user.profile.avatar}

// Array indexing
${items[0].name}

// Arithmetic
${price * quantity}
${total + shipping}
${(subtotal * taxRate) / 100}

// Comparison
${count > 0}
${status === 'active'}
${age >= 18 && age <= 65}

// Ternary
${isLoggedIn ? 'Logout' : 'Login'}
${count === 1 ? 'item' : 'items'}

// String interpolation
${`${firstName} ${lastName}`}
${"Order #" + orderId}

// Null coalescing
${userName ?? 'Anonymous'}
${settings.theme ?? 'light'}

// Array methods
${items.length}
${items.filter(i => i.active).length}
${items.map(i => i.name).join(', ')}

// Built-in functions
${Math.round(price)}
${formatCurrency(amount)}
${formatDate(createdAt, 'MMM DD')}
```

6.6 Built-in Functions

Listing 51: Built-in Functions

```
// Math
Math.round(n)
Math.floor(n)
Math.ceil(n)
Math.min(a, b)
Math.max(a, b)
Math.abs(n)
Math.random()

// String
toLowerCase(s)
toUpperCase(s)
trim(s)
substring(s, start, end)
```

```
replace(s, search, replace)
split(s, delimiter)

// Array
length(arr)
first(arr)
last(arr)
join(arr, delimiter)
includes(arr, value)
indexOf(arr, value)
slice(arr, start, end)

// Date
now()                          // Current timestamp
formatDate(date, format)       // Format date string
addDays(date, n)
diffDays(date1, date2)

// Formatting
formatCurrency(n, currency)    // $1,234.56
formatNumber(n, decimals)      // 1,234.56
formatPercent(n)               // 12.5%

// Utility
generateId()                   // Random unique ID
clamp(n, min, max)             // Constrain to range
```

6.7 Reactive Updates

When a variable changes, all bound properties automatically update:

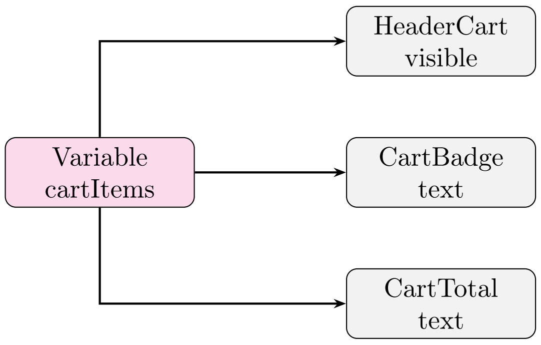

Figure 4: Reactive Variable Binding

7 Conditional Logic

Conditions

Conditions gate whether actions execute based on variable values and expressions.

7.1 Condition Operators

Operator	Symbol	Description
`equals`	`===`	Strict equality
`notEquals`	`!==`	Strict inequality
`greaterThan`	`>`	Greater than
`greaterThanOrEqual`	`>=`	Greater than or equal
`lessThan`	`<`	Less than
`lessThanOrEqual`	`<=`	Less than or equal
`contains`	—	String/array contains value
`notContains`	—	String/array does not contain
`startsWith`	—	String starts with
`endsWith`	—	String ends with
`isEmpty`	—	String/array is empty
`isNotEmpty`	—	String/array is not empty
`isNull`	—	Value is null/undefined
`isNotNull`	—	Value is not null/undefined
`matches`	—	Regex pattern match

Table 20: Condition Operators

7.2 Condition Definition

Listing 52: Condition Examples

```
// Simple condition
{
  "variable": "isLoggedIn",
  "operator": "equals",
  "value": true
}

// Numeric comparison
{
  "variable": "cartItems.length",
  "operator": "greaterThan",
  "value": 0
}

// String comparison
{
  "variable": "userRole",
  "operator": "equals",
  "value": "admin"
}

// Contains check
{
  "variable": "permissions",
  "operator": "contains",
  "value": "edit"
}

// Regex match
{
  "variable": "email",
  "operator": "matches",
  "value": "^[a-zA-Z0-9+_.-]+@[a-zA-Z0-9.-]+$"
}
```

7.3 Compound Conditions

Combine multiple conditions with AND/OR logic:

Listing 53: Compound Conditions

```
// AND - all must be true
{
  "type": "and",
  "conditions": [
    { "variable": "isLoggedIn", "operator": "equals", "value": true },
    { "variable": "hasPermission", "operator": "equals", "value": true }
  ]
}

// OR - any must be true
{
  "type": "or",
  "conditions": [
    { "variable": "userRole", "operator": "equals", "value": "admin" },
    { "variable": "userRole", "operator": "equals", "value": "editor" }
  ]
}

// Nested conditions
{
  "type": "and",
  "conditions": [
    { "variable": "isLoggedIn", "operator": "equals", "value": true },
    {
      "type": "or",
      "conditions": [
        { "variable": "plan", "operator": "equals", "value": "pro" },
        { "variable": "plan", "operator": "equals", "value": "enterprise" }
      ]
    }
  ]
}

// NOT - negate condition
{
  "type": "not",
  "condition": {
    "variable": "isGuest",
    "operator": "equals",
    "value": true
  }
}
```

7.4 Conditional Actions

Execute different actions based on conditions:

Listing 54: Conditional Action Execution

```
{
  "trigger": "onClick",
  "conditionalActions": [
    {
      "condition": {
        "variable": "isLoggedIn",
        "operator": "equals",
        "value": true
      },
      "actions": [
        { "type": "navigate", "destination": "frame:dashboard" }
      ]
    },
    {
      "condition": {
        "variable": "isLoggedIn",
        "operator": "equals",
        "value": false
      },
      "actions": [
        { "type": "openOverlay", "overlay": "component:login-modal" }
      ]
    }
  ]
}

// Simplified if/else
{
  "trigger": "onClick",
  "if": {
    "condition": { "variable": "formValid", "operator": "equals", "value": true },
    "then": [
      { "type": "submitForm", "form": "element:contact-form" }
    ],
    "else": [
      { "type": "animate", "target": "self", "animation": "shake" },
      { "type": "setVariable", "variable": "showErrors", "value": true }
    ]
  }
}
```

7.5 Conditional Visibility

Control element visibility based on conditions:

Listing 55: Conditional Visibility

```
{
  "component": "AdminPanel",
  "visibility": {
    "condition": {
      "variable": "userRole",
      "operator": "equals",
      "value": "admin"
    },
    "hiddenBehavior": "remove"         // "remove" | "hide" | "disable"
  }
}

{
  "component": "EmptyState",
  "visibility": {
    "condition": {
      "variable": "items.length",
      "operator": "equals",
      "value": 0
    }
  }
}
```

8 Prototype Flows

8.1 Flow Definition

A flow is a connected sequence of frames representing a user journey.

Listing 56: Flow Definition

```
{
  "flow": {
    "id": "checkout-flow",
    "name": "Checkout Process",
    "description": "User completes purchase",
    "startingFrame": "frame:cart",
    "frames": [
      "frame:cart",
      "frame:shipping",
      "frame:payment",
      "frame:confirmation"
    ],
    "device": "iphone-14-pro",
    "orientation": "portrait"
  }
}
```

8.2 Starting Points

Define entry points into the prototype:

Listing 57: Starting Points

```
{
  "startingPoints": [
    {
      "id": "start-home",
      "name": "Home Page",
      "frame": "frame:home",
      "description": "Main entry point"
    },
    {
      "id": "start-login",
      "name": "Login Flow",
      "frame": "frame:login",
      "description": "Authentication flow"
    },
    {
      "id": "start-onboarding",
      "name": "Onboarding",
      "frame": "frame:onboarding-1",
      "description": "New user onboarding",
      "variables": {
        "isNewUser": true
      }
    }
  ]
}
```

8.3 Device Frames

Device	Width	Height	Scale
`iphone-se`	375	667	2x
`iphone-14`	390	844	3x
`iphone-14-pro`	393	852	3x
`iphone-14-pro-max`	430	932	3x
`ipad-mini`	744	1133	2x
`ipad-pro-11`	834	1194	2x
`ipad-pro-12.9`	1024	1366	2x
`android-small`	360	640	2x
`android-medium`	360	800	3x
`android-large`	412	915	3x
`desktop-1280`	1280	800	1x
`desktop-1440`	1440	900	1x
`desktop-1920`	1920	1080	1x
`none`	Custom	Custom	1x

Table 21: Device Frame Presets

8.4 Presentation Mode

Settings for prototype playback:

Listing 58: Presentation Settings

```
{
  "presentation": {
    "background": "#1a1a1a",          // Background behind device
    "showDeviceFrame": true,          // Show device bezel
    "showHotspots": false,            // Highlight interactive areas
    "showCursor": true,
    "cursorStyle": "default",         // "default" | "pointer" | "touch"
    "keyboard": {
      "enabled": true,
      "shortcuts": {
        "restart": "r",
        "back": "Backspace",
        "toggleHotspots": "h"
      }
    },
    "touchIndicator": {
      "enabled": true,                // Show touch ripple
      "color": "rgba(0, 0, 0, 0.2)"
    }
  }
}
```

8.5 Prototype History

Navigation history for back/forward:

Listing 59: History Management

```
// History entry on navigation
{
  "historyEntry": {
    "frame": "frame:product-detail",
    "scrollPosition": { "x": 0, "y": 250 },
    "variables": {
      "productId": "12345"
    },
    "timestamp": 1704412800000
  }
}

// Go back action
{
  "type": "navigate",
  "destination": "back",
  "fallback": "frame:home"             // If no history
}
```

8.6 Deep Linking

URL-based navigation into specific states:

Listing 60: Deep Link Configuration

```
{
  "deepLinks": [
    {
      "path": "/product/:id",
      "frame": "frame:product-detail",
      "variables": {
        "productId": "${params.id}"
      }
    },
    {
      "path": "/checkout",
      "frame": "frame:checkout",
      "condition": {
        "variable": "cartItems.length",
        "operator": "greaterThan",
        "value": 0
      },
      "fallback": "frame:cart"
    }
  ]
}
```

9 UnoCSS Integration

9.1 Transition Utilities

Map animation properties to UnoCSS classes:

Listing 61: UnoCSS Transition Mapping

```
// Animation type -> UnoCSS classes
{
  "smartAnimate": {
    "property": "transition-all",
    "duration": "duration-{ms}",       // duration-200, duration-300, etc.
    "easing": "ease-{type}",           // ease-in, ease-out, ease-in-out
    "delay": "delay-{ms}"              // delay-100, delay-200, etc.
  }
}

// Example output:
// class="transition-all duration-300 ease-in-out"
// class="transition-transform duration-200 ease-out"
// class="transition-opacity duration-150 ease-in delay-100"
```

9.2 State Variant Classes

Listing 62: State Variant Classes

```
// Interactive state variants
{
  "states": {
    "hover": "hover:",                 // hover:bg-blue-600
    "focus": "focus:",                 // focus:ring-2
    "focus-visible": "focus-visible:", // focus-visible:outline-2
    "active": "active:",               // active:bg-blue-700
    "disabled": "disabled:",           // disabled:opacity-50
    "checked": "checked:",             // checked:bg-blue-500
    "group-hover": "group-hover:",     // group-hover:text-blue-500
    "peer-checked": "peer-checked:"    // peer-checked:bg-green-500
  }
}

// Button example with all states:
class="
  bg-blue-500 text-white px-4 py-2 rounded-lg
  transition-all duration-200
  hover:bg-blue-600 hover:shadow-lg hover:-translate-y-0.5
  focus:outline-none focus:ring-2 focus:ring-blue-500 focus:ring-offset-2
  active:bg-blue-700 active:translate-y-0 active:shadow-sm
  disabled:opacity-50 disabled:cursor-not-allowed disabled:hover:bg-blue-500
"
```

9.3 Animation Utilities

Listing 63: UnoCSS Animation Utilities

```
// Built-in animations
animate-spin          // Continuous rotation
animate-ping          // Ping/pulse effect
animate-pulse         // Opacity pulse
animate-bounce        // Bounce effect

// Custom animations via config
// uno.config.ts
{
  theme: {
    animation: {
      keyframes: {
        'fade-in': '{ from { opacity: 0; } to { opacity: 1; } }',
        'slide-up': '{ from { transform: translateY(10px); opacity: 0; } to {
           transform: translateY(0); opacity: 1; } }',
        'scale-in': '{ from { transform: scale(0.95); opacity: 0; } to { transform
           : scale(1); opacity: 1; } }',
      },
      durations: {
        'fade-in': '200ms',
        'slide-up': '300ms',
        'scale-in': '200ms',
      },
      timingFns: {
        'fade-in': 'ease-out',
        'slide-up': 'ease-out',
        'scale-in': 'ease-out',
      },
    },
  },
}

// Usage:
class="animate-fade-in"
class="animate-slide-up"
class="animate-scale-in"
```

9.4 Generated CSS Output

Example of complete interactive component CSS:

Listing 64: Generated CSS for Button

```
/* Base styles */
.btn-primary {
  display: inline-flex;
  align-items: center;
  justify-content: center;
  padding: 0.5rem 1rem;
  background-color: #3b82f6;
  color: white;
  border-radius: 0.5rem;
  font-weight: 500;
  transition: all 200ms ease-in-out;
}

/* Hover state */
.btn-primary:hover {
  background-color: #2563eb;
  box-shadow: 0 10px 15px -3px rgba(0, 0, 0, 0.1);
  transform: translateY(-2px);
}

/* Focus state */
.btn-primary:focus {
  outline: none;
  box-shadow: 0 0 0 3px rgba(59, 130, 246, 0.5);
}

/* Active state */
.btn-primary:active {
  background-color: #1d4ed8;
  transform: translateY(0);
  box-shadow: 0 1px 2px 0 rgba(0, 0, 0, 0.05);
}

/* Disabled state */
.btn-primary:disabled {
  opacity: 0.5;
  cursor: not-allowed;
}
.btn-primary:disabled:hover {
  background-color: #3b82f6;
  transform: none;
  box-shadow: none;
}
```

10 Export Specification

10.1 HTML Export

Interactions export as data attributes and inline event handlers:

Listing 65: HTML Export with Interactions

```html
<button
  class="inline-flex items-center justify-center px-4 py-2 bg-blue-500 text-white
      rounded-lg font-medium transition-all duration-200 hover:bg-blue-600 hover:
      shadow-lg hover:-translate-y-0.5 active:bg-blue-700 active:translate-y-0
      focus:outline-none focus:ring-2 focus:ring-blue-500 focus:ring-offset-2
      disabled:opacity-50 disabled:cursor-not-allowed"
  data-interaction="navigate"
  data-destination="checkout.html"
  data-animation="push-left"
  data-duration="400"
>
  Checkout
</button>
```

10.2 JavaScript Export

For frameworks requiring JS interactivity:

Listing 66: JavaScript Interaction Handler

```javascript
// Interaction runtime
const interactions = {
  navigate: (destination, animation, duration) => {
    // Animate out current view
    // Load and animate in new view
  },
  openOverlay: (overlay, position, animation) => {
    // Create backdrop
    // Position overlay
    // Animate in
  },
  setVariable: (name, value) => {
    window.__designLibreVars[name] = value;
    // Trigger reactive updates
  }
};

// Bind to elements
document.querySelectorAll('[data-interaction]').forEach(el => {
  el.addEventListener('click', () => {
    const action = el.dataset.interaction;
    const destination = el.dataset.destination;
    const animation = el.dataset.animation;
    const duration = parseInt(el.dataset.duration);

    interactions[action](destination, animation, duration);
  });
});
```

10.3 React Export

Listing 67: React Component Export

```javascript
import { useState } from 'react';
import { motion, AnimatePresence } from 'framer-motion';
```

```
export function CheckoutButton({ onNavigate }) {
  const [isPressed, setIsPressed] = useState(false);

  return (
    <motion.button
      className="inline-flex items-center justify-center px-4 py-2 bg-blue-500
         text-white rounded-lg font-medium"
      whileHover={{
        backgroundColor: '#2563eb',
        y: -2,
        boxShadow: '0 10px 15px -3px rgba(0, 0, 0, 0.1)'
      }}
      whileTap={{
        backgroundColor: '#1d4ed8',
        y: 0,
        boxShadow: '0 1px 2px 0 rgba(0, 0, 0, 0.05)'
      }}
      transition={{ duration: 0.2 }}
      onClick={() => onNavigate('checkout')}
    >
      Checkout
    </motion.button>
  );
}
```

11 Appendix: Complete Trigger Reference

Trigger	Category	Description
`onClick`	Pointer	Click/tap completes
`onDoubleClick`	Pointer	Double click
`onMouseDown`	Pointer	Pointer pressed
`onMouseUp`	Pointer	Pointer released
`onHover`	Pointer	Pointer enters
`onHoverEnd`	Pointer	Pointer leaves
`onLongPress`	Pointer	Long press/hold
`onRightClick`	Pointer	Context menu
`onDragStart`	Drag	Drag begins
`onDrag`	Drag	During drag
`onDragEnd`	Drag	Drag ends
`onDragEnter`	Drag	Enter drop zone
`onDragLeave`	Drag	Leave drop zone
`onDrop`	Drag	Dropped
`onSwipe`	Gesture	Quick swipe
`onSwipeLeft`	Gesture	Swipe left
`onSwipeRight`	Gesture	Swipe right
`onSwipeUp`	Gesture	Swipe up
`onSwipeDown`	Gesture	Swipe down
`onPinch`	Gesture	Two-finger pinch
`onRotate`	Gesture	Two-finger rotate
`onPan`	Gesture	Single-finger pan
`onFocus`	Focus	Receives focus
`onBlur`	Focus	Loses focus
`onFocusVisible`	Focus	Keyboard focus
`onFocusWithin`	Focus	Child focused
`onKeyDown`	Keyboard	Key pressed
`onKeyUp`	Keyboard	Key released
`onShortcut`	Keyboard	Key combination
`onScroll`	Scroll	Scroll changes
`onScrollStart`	Scroll	Scroll begins
`onScrollEnd`	Scroll	Scroll ends
`onViewportEnter`	Scroll	Enters viewport
`onViewportLeave`	Scroll	Leaves viewport
`onViewportProgress`	Scroll	Visibility changes
`onChange`	Form	Value changes (blur)
`onInput`	Form	Value changes (immediate)
`onSubmit`	Form	Form submitted
`onInvalid`	Form	Validation fails
`onPlay`	Media	Playback starts
`onPause`	Media	Playback pauses
`onEnded`	Media	Playback ends
`onTimeUpdate`	Media	Position changes
`onLoad`	Lifecycle	Component loads

Trigger	Category	Description
`onMount`	Lifecycle	Inserted to DOM
`onUnmount`	Lifecycle	Removed from DOM
`onResize`	Lifecycle	Size changes
`onVisibilityChange`	Lifecycle	Tab visibility
`afterDelay`	Time	After duration
`onInterval`	Time	Repeated interval
`onAnimationEnd`	Time	Animation completes
`onTransitionEnd`	Time	Transition completes
`onVariableChange`	Data	Variable changes
`onConditionMet`	Data	Condition becomes true
`onStateChange`	Data	Component state changes

Table 22: Complete Trigger Reference

"Interactions are the verbs of interface design."