Create a Modal Dialog System with Focus Management and Accessibility

□ Table of Contents

· Next-Generation Modal Features

Table of Contents

Clarify the Problem and Requirements

☐ Back to Top

Problem Understanding

☐ Back to Top

Design a comprehensive modal dialog system that provides accessible, performant, and flexible overlay interfaces for web applications. The system must handle complex focus management, keyboard navigation, screen reader compatibility, and support various modal types while maintaining excellent user experience across devices.

Functional Requirements

☐ Back to Top

- **Modal Types**: Alert, confirm, prompt, custom content, forms, image galleries
- **Focus Management**: Automatic focus trapping, restore previous focus, tab navigation
- **Accessibility**: ARIA compliance, screen reader support, keyboard navigation
- **Stacking**: Multiple modal support, z-index management, modal-over-modal
- **Animations**: Smooth enter/exit transitions, customizable animations
- **Responsive Design**: Mobile-optimized layouts, touch-friendly interactions

- **Portal Rendering**: Render outside DOM hierarchy, avoid z-index conflicts
- **Escape Mechanisms**: ESC key, backdrop click, programmatic close

Non-Functional Requirements

Back to Top

- **Performance**: <16ms render time, 60fps animations, minimal layout shifts
- **Accessibility**: WCAG 2.1 AA compliance, screen reader compatibility
- **Browser Support**: Modern browsers, graceful degradation for older versions
- **Memory Efficiency**: Proper cleanup, no memory leaks, optimized DOM manipulation
- **Customization**: Theming support, flexible layouts, extensible architecture
- **Bundle Size**: <10KB gzipped for core functionality

Key Assumptions

Back to Top

- Maximum concurrent modals: 5 (practical UX limit)
- Animation duration: 200-300ms (optimal perception)
- Focus restoration: Required for accessibility compliance
- Mobile breakpoint: 768px for responsive adaptations
- Browser support: IE11+ (with polyfills), modern browsers native
- Framework agnostic: Core system works with React, Vue, Angular

---

High-Level Architecture

Back to Top

Modal System Architecture

Back to Top

```
graph TB
    subgraph "Application Layer"
        APP_COMPONENTS[Application Components<br/>Forms, buttons, triggers]
        MODAL_TRIGGERS[Modal Triggers<br/>Click handlers, events]
```

```
    BUSINESS_LOGIC[Business Logic<br/>Validation, actions]
end

subgraph "Modal Management"
    MODAL_MANAGER[Modal Manager<br/>Central orchestration]
    MODAL_REGISTRY[Modal Registry<br/>Instance tracking]
    STACK_MANAGER[Stack Manager<br/>Z-index & layering]
    FOCUS_MANAGER[Focus Manager<br/>Accessibility]
end

subgraph "Rendering System"
    PORTAL_MANAGER[Portal Manager<br/>DOM mounting]
    ANIMATION_ENGINE[Animation Engine<br/>Transitions & effects]
    RESPONSIVE_HANDLER[Responsive Handler<br/>Device adaptation]
    THEME_PROVIDER[Theme Provider<br/>Styling system]
end

subgraph "Accessibility Layer"
    ARIA_MANAGER[ARIA Manager<br/>Semantic attributes]
    KEYBOARD_HANDLER[Keyboard Handler<br/>Navigation & shortcuts]
    SCREEN_READER[Screen Reader Support<br/>Announcements]
    FOCUS_TRAP[Focus Trap<br/>Containment logic]
end

subgraph "Event System"
    EVENT_BUS[Event Bus<br/>Modal communications]
    LIFECYCLE_HOOKS[Lifecycle Hooks<br/>Before/after events]
    STATE_MANAGER[State Manager<br/>Modal state tracking]
    CLEANUP_SERVICE[Cleanup Service<br/>Memory management]
end

APP_COMPONENTS --> MODAL_TRIGGERS
MODAL_TRIGGERS --> BUSINESS_LOGIC
BUSINESS_LOGIC --> MODAL_MANAGER

MODAL_MANAGER --> MODAL_REGISTRY
MODAL_MANAGER --> STACK_MANAGER
MODAL_MANAGER --> FOCUS_MANAGER

MODAL_REGISTRY --> PORTAL_MANAGER
STACK_MANAGER --> ANIMATION_ENGINE
FOCUS_MANAGER --> RESPONSIVE_HANDLER

PORTAL_MANAGER --> ARIA_MANAGER
ANIMATION_ENGINE --> KEYBOARD_HANDLER
```

```
    RESPONSIVE_HANDLER --> SCREEN_READER
    THEME_PROVIDER --> FOCUS_TRAP

    ARIA_MANAGER --> EVENT_BUS
    KEYBOARD_HANDLER --> LIFECYCLE_HOOKS
    SCREEN_READER --> STATE_MANAGER
    FOCUS_TRAP --> CLEANUP_SERVICE
```

Focus Management Flow

Back to Top

```
graph TD
    subgraph "Focus Initialization"
        MODAL_OPEN[Modal Opens]
        SAVE_FOCUS[Save Current Focus<br/>Store active element]
        FIND_FOCUSABLE[Find Focusable Elements<br/>Query interactive elements]
        SET_INITIAL_FOCUS[Set Initial Focus<br/>First focusable or specified]
    end

    subgraph "Focus Trap Implementation"
        TAB_INTERCEPT[Tab Intercept<br/>Monitor tab navigation]
        SHIFT_TAB_INTERCEPT[Shift+Tab Intercept<br/>Reverse navigation]
        BOUNDARY_CHECK[Boundary Check<br/>First/last element detection]
        FOCUS_REDIRECT[Focus Redirect<br/>Wrap to opposite end]
    end

    subgraph "Focus Restoration"
        MODAL_CLOSE[Modal Closes]
        RESTORE_FOCUS[Restore Focus<br/>Return to saved element]
        FALLBACK_FOCUS[Fallback Focus<br/>Body if element removed]
        ACCESSIBILITY_ANNOUNCE[Accessibility Announce<br/>Screen reader notification]
    end

    subgraph "Edge Cases"
        NESTED_MODALS[Nested Modals<br/>Stack management]
        DYNAMIC_CONTENT[Dynamic Content<br/>Content changes]
        DISABLED_ELEMENTS[Disabled Elements<br/>Skip non-interactive]
        INVISIBLE_ELEMENTS[Invisible Elements<br/>Skip hidden items]
    end

    MODAL_OPEN --> SAVE_FOCUS
    SAVE_FOCUS --> FIND_FOCUSABLE
    FIND_FOCUSABLE --> SET_INITIAL_FOCUS
```

```
    SET_INITIAL_FOCUS --> TAB_INTERCEPT
    TAB_INTERCEPT --> SHIFT_TAB_INTERCEPT
    SHIFT_TAB_INTERCEPT --> BOUNDARY_CHECK
    BOUNDARY_CHECK --> FOCUS_REDIRECT

    FOCUS_REDIRECT --> MODAL_CLOSE
    MODAL_CLOSE --> RESTORE_FOCUS
    RESTORE_FOCUS --> FALLBACK_FOCUS
    FALLBACK_FOCUS --> ACCESSIBILITY_ANNOUNCE

    TAB_INTERCEPT --> NESTED_MODALS
    FIND_FOCUSABLE --> DYNAMIC_CONTENT
    BOUNDARY_CHECK --> DISABLED_ELEMENTS
    FOCUS_REDIRECT --> INVISIBLE_ELEMENTS
```

UI/UX and Component Structure

☐ Back to Top

Component Architecture

☐ Back to Top

```
graph TD
    subgraph "Core Components"
        MODAL_PROVIDER[Modal Provider<br/>Context & state management]
        MODAL_CONTAINER[Modal Container<br/>Wrapper & positioning]
        MODAL_BACKDROP[Modal Backdrop<br/>Overlay & click handling]
        MODAL_CONTENT[Modal Content<br/>Main content area]
    end

    subgraph "Specialized Modals"
        ALERT_MODAL[Alert Modal<br/>Simple notifications]
        CONFIRM_MODAL[Confirm Modal<br/>Yes/no decisions]
        PROMPT_MODAL[Prompt Modal<br/>Text input dialogs]
        CUSTOM_MODAL[Custom Modal<br/>Flexible content]
        FORM_MODAL[Form Modal<br/>Complex forms]
        GALLERY_MODAL[Gallery Modal<br/>Image/media viewer]
    end
```

```
subgraph "UI Components"
    MODAL_HEADER[Modal Header<br/>Title & close button]
    MODAL_BODY[Modal Body<br/>Scrollable content]
    MODAL_FOOTER[Modal Footer<br/>Action buttons]
    CLOSE_BUTTON[Close Button<br/>Accessible close action]
    LOADING_SPINNER[Loading Spinner<br/>Async operations]
end

subgraph "Accessibility Components"
    FOCUS_TRAP_COMPONENT[Focus Trap<br/>Keyboard navigation]
    ARIA_LIVE_REGION[ARIA Live Region<br/>Status announcements]
    SKIP_LINK[Skip Link<br/>Navigation assistance]
    HIGH_CONTRAST[High Contrast Mode<br/>Visual accessibility]
end

subgraph "Animation Components"
    FADE_TRANSITION[Fade Transition<br/>Opacity animation]
    SCALE_TRANSITION[Scale Transition<br/>Size animation]
    SLIDE_TRANSITION[Slide Transition<br/>Position animation]
    SPRING_ANIMATION[Spring Animation<br/>Physics-based motion]
end

subgraph "Utility Hooks/Services"
    USE_MODAL[useModal Hook<br/>Modal state management]
    USE_FOCUS_TRAP[useFocusTrap Hook<br/>Focus management]
    USE_ESCAPE_KEY[useEscapeKey Hook<br/>Keyboard handling]
    USE_LOCK_SCROLL[useLockScroll Hook<br/>Body scroll prevention]
    USE_PORTAL[usePortal Hook<br/>DOM portal rendering]
end

MODAL_PROVIDER --> MODAL_CONTAINER
MODAL_CONTAINER --> MODAL_BACKDROP
MODAL_BACKDROP --> MODAL_CONTENT

MODAL_CONTENT --> ALERT_MODAL
MODAL_CONTENT --> CONFIRM_MODAL
MODAL_CONTENT --> PROMPT_MODAL
MODAL_CONTENT --> CUSTOM_MODAL
MODAL_CONTENT --> FORM_MODAL
MODAL_CONTENT --> GALLERY_MODAL

MODAL_CONTENT --> MODAL_HEADER
MODAL_CONTENT --> MODAL_BODY
MODAL_CONTENT --> MODAL_FOOTER
```

```
    MODAL_HEADER --> CLOSE_BUTTON
    MODAL_BODY --> LOADING_SPINNER

    MODAL_CONTAINER --> FOCUS_TRAP_COMPONENT
    MODAL_CONTAINER --> ARIA_LIVE_REGION
    MODAL_CONTAINER --> SKIP_LINK
    MODAL_CONTAINER --> HIGH_CONTRAST

    MODAL_BACKDROP --> FADE_TRANSITION
    MODAL_CONTENT --> SCALE_TRANSITION
    MODAL_CONTENT --> SLIDE_TRANSITION
    MODAL_CONTENT --> SPRING_ANIMATION

    ALERT_MODAL --> USE_MODAL
    CONFIRM_MODAL --> USE_FOCUS_TRAP
    PROMPT_MODAL --> USE_ESCAPE_KEY
    CUSTOM_MODAL --> USE_LOCK_SCROLL
    FORM_MODAL --> USE_PORTAL
```

React Component Implementation ☐ Back to Top

ModalProvider.jsx

```jsx
import React, { createContext, useContext, useState, useCallback } from 'react';
import ModalContainer from './ModalContainer';

const ModalContext = createContext();

export const useModal = () => {
  const context = useContext(ModalContext);
  if (!context) {
    throw new Error('useModal must be used within a ModalProvider');
  }
  return context;
};

export const ModalProvider = ({ children }) => {
  const [modals, setModals] = useState([]);

  const openModal = useCallback((modalComponent, props = {}) => {
    const id = Date.now().toString();
    const modal = {
      id,
      component: modalComponent,
```

```
      props: {
        ...props,
        id,
        onClose: () => closeModal(id)
      }
    };

    setModals(prev => [...prev, modal]);
    return id;
  }, []);

  const closeModal = useCallback((id) => {
    setModals(prev => prev.filter(modal => modal.id !== id));
  }, []);

  const alert = useCallback((message, title = 'Alert') => {
    return openModal('AlertModal', { message, title });
  }, [openModal]);

  const confirm = useCallback((message, title = 'Confirm') => {
    return new Promise((resolve) => {
      openModal('ConfirmModal', {
        message,
        title,
        onConfirm: () => resolve(true),
        onCancel: () => resolve(false)
      });
    });
  }, [openModal]);

  return (
    <ModalContext.Provider value={{
      modals,
      openModal,
      closeModal,
      alert,
      confirm
    }}>
      {children}
      <ModalContainer />
    </ModalContext.Provider>
  );
};
```

Modal.jsx

```jsx
import React, { useEffect, useRef, useCallback } from 'react';
import { useFocusTrap } from './hooks/useFocusTrap';
import { useEscapeKey } from './hooks/useEscapeKey';
import { useLockScroll } from './hooks/useLockScroll';

const Modal = ({
  isOpen,
  onClose,
  children,
  title,
  size = 'medium',
  closable = true,
  closeOnBackdrop = true,
  closeOnEscape = true,
  animation = 'fade',
  className = ''
}) => {
  const modalRef = useRef(null);
  const backdropRef = useRef(null);

  useFocusTrap(modalRef, isOpen);
  useEscapeKey(isOpen && closeOnEscape ? onClose : null);
  useLockScroll(isOpen);

  const handleBackdropClick = useCallback((e) => {
    if (closeOnBackdrop && e.target === backdropRef.current) {
      onClose();
    }
  }, [closeOnBackdrop, onClose]);

  if (!isOpen) return null;

  return (
    <div
      ref={backdropRef}
      className={`modal-backdrop ${animation}`}
      onClick={handleBackdropClick}
      role="presentation"
    >
      <div
        ref={modalRef}
        className={`modal ${size} ${className}`}
        role="dialog"
        aria-modal="true"
        aria-labelledby={title ? 'modal-title' : undefined}
```

```
      >
        {title && (
          <div className="modal-header">
            <h2 id="modal-title" className="modal-title">
              {title}
            </h2>
            {closable && (
              <button
                className="modal-close-button"
                onClick={onClose}
                aria-label="Close modal"
              >
                ×
              </button>
            )}
          </div>
        )}

        <div className="modal-body">
          {children}
        </div>
      </div>
    </div>
  );
};

export default Modal;
```

Custom Hooks

```
// hooks/useFocusTrap.js
import { useEffect } from 'react';

export const useFocusTrap = (containerRef, isActive) => {
  useEffect(() => {
    if (!isActive || !containerRef.current) return;

    const container = containerRef.current;
    const focusableElements = container.querySelectorAll(
      'button, [href], input, select, textarea, [tabindex]:not([tabindex="-1"])'
    );

    const firstElement = focusableElements[0];
    const lastElement = focusableElements[focusableElements.length - 1];

    const handleTabKey = (e) => {
```

```
      if (e.key !== 'Tab') return;

      if (e.shiftKey) {
        if (document.activeElement === firstElement) {
          e.preventDefault();
          lastElement.focus();
        }
      } else {
        if (document.activeElement === lastElement) {
          e.preventDefault();
          firstElement.focus();
        }
      }
    };

    document.addEventListener('keydown', handleTabKey);
    return () => document.removeEventListener('keydown', handleTabKey);
  }, [isActive, containerRef]);
};

// hooks/useEscapeKey.js
export const useEscapeKey = (callback) => {
  useEffect(() => {
    if (!callback) return;

    const handleEscape = (e) => {
      if (e.key === 'Escape') {
        callback();
      }
    };

    document.addEventListener('keydown', handleEscape);
    return () => document.removeEventListener('keydown', handleEscape);
  }, [callback]);
};

// hooks/useLockScroll.js
export const useLockScroll = (isLocked) => {
  useEffect(() => {
    if (isLocked) {
      const originalStyle = window.getComputedStyle(document.body).overflow;
      document.body.style.overflow = 'hidden';

      return () => {
        document.body.style.overflow = originalStyle;
```

```
      };
    }
  }, [isLocked]);
};
```

Responsive Modal Layout

☐ Back to Top

```
graph TD
    subgraph "Mobile Layout (< 768px)"
        M_FULLSCREEN[Full-screen Modal<br/>Maximum viewport usage]
        M_BOTTOM_SHEET[Bottom Sheet<br/>Slide-up interaction]
        M_TOUCH_TARGETS[Large Touch Targets<br/>44px minimum]
        M_SWIPE_DISMISS[Swipe to Dismiss<br/>Gesture support]
        M_SAFE_AREAS[Safe Area Handling<br/>Notch accommodation]
    end

    subgraph "Tablet Layout (768px - 1024px)"
        T_CENTERED[Centered Modal<br/>With backdrop]
        T_MAX_WIDTH[Max Width 80%<br/>Readable content width]
        T_TOUCH_KEYBOARD[Touch + Keyboard<br/>Hybrid interaction]
        T_ADAPTIVE_SIZE[Adaptive Sizing<br/>Content-based height]
        T_LANDSCAPE[Landscape Mode<br/>Horizontal optimization]
    end

    subgraph "Desktop Layout (> 1024px)"
        D_OVERLAY[Overlay Modal<br/>Traditional dialog]
        D_FIXED_SIZE[Fixed Dimensions<br/>Consistent sizing]
        D_KEYBOARD_NAV[Keyboard Navigation<br/>Full accessibility]
        D_MULTIPLE_MODALS[Multiple Modals<br/>Stacking support]
        D_DRAG_RESIZE[Drag & Resize<br/>User control]
    end

    M_FULLSCREEN --> T_CENTERED
    M_BOTTOM_SHEET --> T_MAX_WIDTH
    M_TOUCH_TARGETS --> T_TOUCH_KEYBOARD
    M_SWIPE_DISMISS --> T_ADAPTIVE_SIZE
    M_SAFE_AREAS --> T_LANDSCAPE

    T_CENTERED --> D_OVERLAY
    T_MAX_WIDTH --> D_FIXED_SIZE
    T_TOUCH_KEYBOARD --> D_KEYBOARD_NAV
    T_ADAPTIVE_SIZE --> D_MULTIPLE_MODALS
```

```
    T_LANDSCAPE --> D_DRAG_RESIZE
```

Real-Time Sync, Data Modeling & APIs

☐ Back to Top

Modal State Management

☐ Back to Top

State Machine Implementation ☐ Back to Top

```
stateDiagram-v2
    [*] --> Closed
    Closed --> Opening: open()
    Opening --> Open: Animation complete
    Open --> Closing: close()
    Closing --> Closed: Animation complete

    Open --> Loading: Async operation
    Loading --> Open: Operation complete
    Loading --> Error: Operation failed
    Error --> Open: Retry/dismiss

    Open --> Minimized: minimize()
    Minimized --> Open: restore()
    Minimized --> Closing: close()

    note right of Opening
        - Mount component
        - Start enter animation
        - Set up focus trap
        - Lock body scroll
    end note

    note right of Open
        - Focus management active
        - Keyboard navigation enabled
```

```
        - Escape key handling
        - Backdrop click handling
    end note

    note right of Closing
        - Start exit animation
        - Restore focus
        - Unlock body scroll
        - Clean up listeners
    end note
```

Focus Trap Algorithm

Back to Top

Advanced Focus Management Back to Top

```
graph TD
    A[Initialize Focus Trap] --> B[Query Focusable Elements<br/>Selectors: input, button
    B --> C[Filter Visible Elements<br/>Check offsetParent, visibility, display]
    C --> D[Sort by Tab Index<br/>Positive tabindex first, then DOM order]
    D --> E[Set Initial Focus<br/>data-autofocus or first element]

    E --> F[Listen for Tab Key]
    F --> G{Tab Direction}
    G -->|Forward Tab| H[Current Index + 1]
    G -->|Shift + Tab| I[Current Index - 1]

    H --> J{At End?}
    I --> K{At Start?}

    J -->|Yes| L[Focus First Element]
    J -->|No| M[Focus Next Element]

    K -->|Yes| N[Focus Last Element]
    K -->|No| O[Focus Previous Element]

    L --> P[Prevent Default Event]
    M --> P
    N --> P
    O --> P
```

```
    P --> Q[Update Current Index]
    Q --> F

    subgraph "Edge Cases"
        R[Dynamic Content Changes]
        S[Programmatic Focus Changes]
        T[Elements Becoming Disabled]
        U[Nested Interactive Elements]
    end

    B --> R
    C --> S
    D --> T
    E --> U
```

Animation System

☐ Back to Top

Physics-Based Animations ☐ Back to Top

```
graph TD
    subgraph "Animation Configuration"
        SPRING_CONFIG[Spring Configuration<br/>Tension, friction, mass]
        EASING_FUNCTIONS[Easing Functions<br/>Bezier curves, presets]
        DURATION_CALC[Duration Calculation<br/>Content-based timing]
        PERFORMANCE_MODE[Performance Mode<br/>Reduced motion support]
    end

    subgraph "Animation States"
        ENTER_ANIMATION[Enter Animation<br/>Scale + opacity + transform]
        EXIT_ANIMATION[Exit Animation<br/>Reverse enter sequence]
        LOADING_ANIMATION[Loading Animation<br/>Skeleton + spinner]
        ERROR_ANIMATION[Error Animation<br/>Shake + color change]
    end

    subgraph "Performance Optimization"
        GPU_ACCELERATION[GPU Acceleration<br/>Transform3d, will-change]
        RAF_SCHEDULING[RAF Scheduling<br/>Frame-perfect timing]
        COMPOSITE_LAYERS[Composite Layers<br/>Isolated rendering]
        MEMORY_CLEANUP[Memory Cleanup<br/>Animation completion]
    end
```

```
    subgraph "Accessibility Integration"
        REDUCED_MOTION[Reduced Motion<br/>prefers-reduced-motion]
        FOCUS_TIMING[Focus Timing<br/>Animation + focus sync]
        SCREEN_READER_COMPAT[Screen Reader Compat<br/>Announcement timing]
        HIGH_CONTRAST_MODE[High Contrast<br/>Animation simplification]
    end

    SPRING_CONFIG --> ENTER_ANIMATION
    EASING_FUNCTIONS --> EXIT_ANIMATION
    DURATION_CALC --> LOADING_ANIMATION
    PERFORMANCE_MODE --> ERROR_ANIMATION

    ENTER_ANIMATION --> GPU_ACCELERATION
    EXIT_ANIMATION --> RAF_SCHEDULING
    LOADING_ANIMATION --> COMPOSITE_LAYERS
    ERROR_ANIMATION --> MEMORY_CLEANUP

    GPU_ACCELERATION --> REDUCED_MOTION
    RAF_SCHEDULING --> FOCUS_TIMING
    COMPOSITE_LAYERS --> SCREEN_READER_COMPAT
    MEMORY_CLEANUP --> HIGH_CONTRAST_MODE
```

Data Models

☐ Back to Top

Modal Configuration Schema ☐ Back to Top

```typescript
interface ModalConfig {
  id: string
  type: 'alert' | 'confirm' | 'prompt' | 'custom'

  // Content
  title?: string
  message?: string
  content?: ReactNode

  // Behavior
  closable: boolean
  closeOnEscape: boolean
  closeOnBackdrop: boolean
```

```
  persistent: boolean

  // Styling
  size: 'small' | 'medium' | 'large' | 'fullscreen'
  theme: 'light' | 'dark' | 'auto'
  className?: string
  style?: CSSProperties

  // Animation
  animation: {
    type: 'fade' | 'scale' | 'slide' | 'spring'
    duration: number
    easing: string
    reducedMotion?: boolean
  }

  // Accessibility
  accessibility: {
    role: string
    labelledBy?: string
    describedBy?: string
    autoFocus?: boolean | string
    restoreFocus: boolean
    announceOpen?: string
    announceClose?: string
  }

  // Callbacks
  onOpen?: () => void
  onClose?: (reason: CloseReason) => void
  onConfirm?: () => void
  onCancel?: () => void

  // Stack management
  zIndex?: number
  level: number
  parent?: string
}
```

Focus Management State □ Back to Top

```
interface FocusState {
  activeModal: string | null
```

```
  focusStack: Array<{
    modalId: string
    previousFocus: HTMLElement | null
    focusableElements: HTMLElement[]
    currentIndex: number
    restoreElement: HTMLElement | null
  }>
  trapEnabled: boolean
  lastUserAction: 'keyboard' | 'mouse' | 'touch'
}
```

Performance and Scalability

☐ Back to Top

Memory Management

☐ Back to Top

Component Lifecycle Optimization ☐ Back to Top

```
graph TD
    A[Modal Mount] --> B[Initialize State<br/>Minimal initial render]
    B --> C[Portal Creation<br/>Create DOM mount point]
    C --> D[Event Listener Setup<br/>Global keyboard, resize]
    D --> E[Animation Setup<br/>CSS class preparation]

    E --> F[Content Render<br/>Main modal content]
    F --> G[Focus Management<br/>Trap activation]
    G --> H[Accessibility Setup<br/>ARIA attributes]

    H --> I[Modal Active State<br/>User interaction ready]

    I --> J[Modal Close Triggered]
    J --> K[Animation Cleanup<br/>Remove CSS classes]
    K --> L[Focus Restoration<br/>Return to previous element]
    L --> M[Event Cleanup<br/>Remove global listeners]
    M --> N[Portal Cleanup<br/>Remove DOM mount point]
```

```
    N --> O[Component Unmount<br/>Memory release]

    subgraph "Memory Optimization"
        P[Weak References<br/>Avoid circular refs]
        Q[Event Delegation<br/>Minimize listeners]
        R[Lazy Loading<br/>Content on demand]
        S[Cleanup Timeouts<br/>Clear pending timers]
    end

    D --> P
    F --> Q
    G --> R
    K --> S
```

Performance Monitoring

☐ Back to Top

Real-time Performance Metrics ☐ Back to Top

```
graph TB
    subgraph "Rendering Performance"
        RENDER_TIME[Render Time<br/>Component mount to paint]
        ANIMATION_FPS[Animation FPS<br/>Transition smoothness]
        LAYOUT_SHIFTS[Layout Shifts<br/>CLS measurement]
        PAINT_TIMING[Paint Timing<br/>First/largest contentful paint]
    end

    subgraph "Interaction Performance"
        FOCUS_LATENCY[Focus Latency<br/>Tab navigation response]
        KEYBOARD_RESPONSE[Keyboard Response<br/>Event handler execution]
        TOUCH_RESPONSE[Touch Response<br/>Mobile interaction timing]
        ACCESSIBILITY_TIMING[Accessibility Timing<br/>Screen reader delays]
    end

    subgraph "Memory Metrics"
        COMPONENT_COUNT[Component Count<br/>Active modal instances]
        EVENT_LISTENERS[Event Listeners<br/>Global listener count]
        DOM_NODES[DOM Nodes<br/>Modal-related elements]
        MEMORY_LEAKS[Memory Leaks<br/>Unreleased references]
    end
```

```
    subgraph "User Experience"
        MODAL_BOUNCE_RATE[Modal Bounce Rate<br/>Immediate closures]
        COMPLETION_RATE[Completion Rate<br/>Successful actions]
        ERROR_RATE[Error Rate<br/>Failed operations]
        ACCESSIBILITY_SCORE[Accessibility Score<br/>WCAG compliance]
    end

    RENDER_TIME --> MODAL_BOUNCE_RATE
    ANIMATION_FPS --> COMPLETION_RATE
    FOCUS_LATENCY --> ERROR_RATE
    KEYBOARD_RESPONSE --> ACCESSIBILITY_SCORE
```

Security and Privacy

☐ Back to Top

Security Considerations

☐ Back to Top

Content Security and XSS Prevention ☐ Back to Top

```
graph TD
    subgraph "Input Sanitization"
        CONTENT_VALIDATION[Content Validation<br/>HTML sanitization]
        XSS_PREVENTION[XSS Prevention<br/>Script tag filtering]
        URL_VALIDATION[URL Validation<br/>Safe link checking]
        FILE_VALIDATION[File Validation<br/>Upload security]
    end

    subgraph "DOM Security"
        PORTAL_ISOLATION[Portal Isolation<br/>Separate render context]
        EVENT_SANITIZATION[Event Sanitization<br/>Handler validation]
        ATTRIBUTE_FILTERING[Attribute Filtering<br/>Dangerous attr removal]
        SCRIPT_ISOLATION[Script Isolation<br/>Execution context control]
    end

    subgraph "Access Control"
```

```
        PERMISSION_CHECK[Permission Check<br/>User authorization]
        ROLE_VALIDATION[Role Validation<br/>Feature access control]
        SESSION_VALIDATION[Session Validation<br/>Authentication status]
        RATE_LIMITING[Rate Limiting<br/>Abuse prevention]
    end

    CONTENT_VALIDATION --> PORTAL_ISOLATION
    XSS_PREVENTION --> EVENT_SANITIZATION
    URL_VALIDATION --> ATTRIBUTE_FILTERING
    FILE_VALIDATION --> SCRIPT_ISOLATION

    PORTAL_ISOLATION --> PERMISSION_CHECK
    EVENT_SANITIZATION --> ROLE_VALIDATION
    ATTRIBUTE_FILTERING --> SESSION_VALIDATION
    SCRIPT_ISOLATION --> RATE_LIMITING
```

Testing, Monitoring, and Maintainability

☐ Back to Top

Comprehensive Testing Strategy

☐ Back to Top

Multi-Layer Testing Approach ☐ Back to Top

```
graph TD
    subgraph "Unit Tests"
        UT1[Focus Management Tests<br/>Tab navigation logic]
        UT2[Animation Tests<br/>Transition states]
        UT3[State Management Tests<br/>Modal lifecycle]
        UT4[Accessibility Tests<br/>ARIA attributes]
    end

    subgraph "Integration Tests"
        IT1[Component Integration<br/>Parent-child communication]
        IT2[Event Handling<br/>Keyboard, mouse, touch]
        IT3[Portal Rendering<br/>DOM manipulation]
```

```
    IT4[Theme Integration<br/>Style application]
end

subgraph "E2E Tests"
    E2E1[User Journey Tests<br/>Complete workflows]
    E2E2[Cross-browser Tests<br/>Browser compatibility]
    E2E3[Device Testing<br/>Mobile, tablet, desktop]
    E2E4[Performance Tests<br/>Load time, animation fps]
end

subgraph "Accessibility Tests"
    AT1[Screen Reader Tests<br/>NVDA, JAWS, VoiceOver]
    AT2[Keyboard Navigation<br/>Tab order, shortcuts]
    AT3[Color Contrast<br/>WCAG compliance]
    AT4[Focus Management<br/>Visual indicators]
end

UT1 --> IT1
UT2 --> IT2
UT3 --> IT3
UT4 --> IT4

IT1 --> E2E1
IT2 --> E2E2
IT3 --> E2E3
IT4 --> E2E4

E2E1 --> AT1
E2E2 --> AT2
E2E3 --> AT3
E2E4 --> AT4
```

Trade-offs, Deep Dives, and Extensions

☐ Back to Top

Modal vs Alternative Patterns

☐ Back to Top

Pattern	Modal Dialog	Slide Panel	Inline Expansion	New Page
Context Preservation	Excellent	Good	Excellent	Poor
Mobile Experience	Good	Excellent	Good	Excellent
Accessibility	Complex	Moderate	Simple	Simple
Performance	Good	Good	Excellent	Variable
SEO Impact	None	None	None	Positive
Deep Linking	Difficult	Difficult	Possible	Natural

Advanced Features

☐ Back to Top

Intelligent Modal Behavior ☐ Back to Top

```
graph TD
    subgraph "Adaptive Behavior"
        DEVICE_DETECTION[Device Detection<br/>Mobile, tablet, desktop]
        USAGE_PATTERNS[Usage Patterns<br/>User behavior analysis]
        CONTEXT_AWARENESS[Context Awareness<br/>Page state, user flow]
        PERFORMANCE_ADAPTATION[Performance Adaptation<br/>Hardware capabilities]
    end

    subgraph "Smart Positioning"
        VIEWPORT_ANALYSIS[Viewport Analysis<br/>Available space calculation]
        COLLISION_DETECTION[Collision Detection<br/>Overlap avoidance]
        SMART_SIZING[Smart Sizing<br/>Content-aware dimensions]
        MULTI_SCREEN[Multi-screen Support<br/>Display awareness]
    end

    subgraph "Predictive Features"
        PRELOAD_CONTENT[Preload Content<br/>Anticipated modals]
        GESTURE_PREDICTION[Gesture Prediction<br/>Touch pattern learning]
        ACCESSIBILITY_PREDICTION[Accessibility Prediction<br/>User needs detection]
        PERFORMANCE_PREDICTION[Performance Prediction<br/>Resource planning]
    end

    DEVICE_DETECTION --> VIEWPORT_ANALYSIS
    USAGE_PATTERNS --> COLLISION_DETECTION
    CONTEXT_AWARENESS --> SMART_SIZING
    PERFORMANCE_ADAPTATION --> MULTI_SCREEN
```

```
VIEWPORT_ANALYSIS --> PRELOAD_CONTENT
COLLISION_DETECTION --> GESTURE_PREDICTION
SMART_SIZING --> ACCESSIBILITY_PREDICTION
MULTI_SCREEN --> PERFORMANCE_PREDICTION
```

Future Extensions

□ Back to Top

Next-Generation Modal Features □ Back to Top

---

1. **Voice Integration**:
   - Voice-controlled modal navigation
   - Speech-to-text for form inputs
   - Audio feedback for actions
   - Voice accessibility features
2. **Gesture Recognition**:
   - Touch gesture controls
   - Eye tracking navigation
   - Hand gesture detection
   - Spatial interaction support
3. **AI-Powered UX**:
   - Intent prediction for modal content
   - Adaptive layouts based on usage
   - Personalized interaction patterns
   - Intelligent focus management
4. **Immersive Technologies**:
   - AR/VR modal overlays
   - 3D spatial positioning
   - Haptic feedback integration
   - Immersive interaction paradigms

This comprehensive design provides a robust foundation for building an accessible, performant, and flexible modal dialog system that handles complex focus management, provides excellent user experience across all devices, and maintains high accessibility standards while being extensible for future enhancements.

TypeScript Interfaces & Component Props

□ Back to Top

Core Data Interfaces

```typescript
interface ModalConfig {
  id: string;
  type: 'alert' | 'confirm' | 'prompt' | 'custom';
  title?: string;
  content: React.ReactNode | string;
  size: 'sm' | 'md' | 'lg' | 'xl' | 'fullscreen';
  position: 'center' | 'top' | 'bottom' | 'custom';
  backdrop: boolean | 'static';
  keyboard: boolean;
  animation: 'fade' | 'slide' | 'zoom' | 'none';
  zIndex?: number;
}

interface ModalAction {
  id: string;
  label: string;
  variant: 'primary' | 'secondary' | 'danger' | 'success';
  onClick: (modal: ModalInstance) => void | Promise<void>;
  disabled?: boolean;
  loading?: boolean;
  autoClose?: boolean;
}

interface ModalInstance {
  id: string;
  config: ModalConfig;
  isOpen: boolean;
  isAnimating: boolean;
  openedAt: Date;
  data?: any;
  resolve?: (value: any) => void;
  reject?: (reason: any) => void;
}

interface ModalState {
  instances: ModalInstance[];
  activeModalId?: string;
  maxZIndex: number;
  focusHistory: HTMLElement[];
  bodyScrollY: number;
  isBodyScrollLocked: boolean;
}
```

```typescript
interface DialogResult<T = any> {
  confirmed: boolean;
  data?: T;
  action?: string;
}
```

Component Props Interfaces

```typescript
interface ModalProps {
  isOpen: boolean;
  onClose: () => void;
  title?: string;
  size?: ModalSize;
  backdrop?: boolean | 'static';
  keyboard?: boolean;
  animation?: AnimationType;
  className?: string;
  style?: React.CSSProperties;
  children: React.ReactNode;
  zIndex?: number;
}

interface ModalHeaderProps {
  title?: string;
  showCloseButton?: boolean;
  onClose?: () => void;
  className?: string;
  children?: React.ReactNode;
}

interface ModalBodyProps {
  className?: string;
  scrollable?: boolean;
  maxHeight?: string;
  children: React.ReactNode;
}

interface ModalFooterProps {
  actions?: ModalAction[];
  align?: 'left' | 'center' | 'right' | 'space-between';
  className?: string;
  children?: React.ReactNode;
}

interface ConfirmDialogProps {
```

```
  title?: string;
  message: string;
  confirmText?: string;
  cancelText?: string;
  variant?: 'default' | 'danger';
  onConfirm?: () => void | Promise<void>;
  onCancel?: () => void;
}

interface PromptDialogProps {
  title?: string;
  message: string;
  placeholder?: string;
  defaultValue?: string;
  validator?: (value: string) => string | null;
  onSubmit?: (value: string) => void | Promise<void>;
  onCancel?: () => void;
}
```

API Reference

☐ Back to Top

Modal Management

- `ModalService.open(config)` - Open new modal with configuration options
- `ModalService.close(modalId?)` - Close specific modal or topmost modal
- `ModalService.closeAll()` - Close all open modals in stack
- `ModalService.update(modalId, config)` - Update modal configuration dynamically
- `ModalService.isOpen(modalId?)` - Check if modal is open by ID or any modal

Dialog Utilities

- `ModalService.alert(message, options?)` - Show alert dialog with OK button
- `ModalService.confirm(message, options?)` - Show confirmation dialog with Yes/No
- `ModalService.prompt(message, options?)` - Show input prompt dialog
- `ModalService.custom(component, props?)` - Open custom modal component
- `ModalService.loading(message?, options?)` - Show loading modal with spinner

Stack Management

- `ModalService.getStack()` - Get current modal stack information
- `ModalService.setMaxModals(limit)` - Set maximum concurrent modals allowed
- `ModalService.bringToFront(modalId)` - Bring specific modal to front of stack
- `ModalService.sendToBack(modalId)` - Send modal to back of stack
- `ModalService.getActiveModal()` - Get currently active (topmost) modal

Event Management

- `ModalService.on(event, callback)` - Subscribe to modal events
- `ModalService.off(event, callback)` - Unsubscribe from modal events
- `ModalService.emit(event, data)` - Emit custom modal event
- Events: `'open' | 'close' | 'beforeClose' | 'afterOpen' | 'stackChange'`

Configuration

- `ModalService.setDefaults(config)` - Set default modal configuration
- `ModalService.getDefaults()` - Get current default configuration
- `ModalService.registerAnimation(name, config)` - Register custom animation
- `ModalService.setTheme(theme)` - Apply theme to all modals
- `ModalService.resetConfig()` - Reset to factory default configuration

Accessibility

- `ModalService.setA11yConfig(config)` - Configure accessibility options
- `ModalService.enableFocusTrap(modalId)` - Enable focus trapping for modal
- `ModalService.disableFocusTrap(modalId)` - Disable focus trapping
- `ModalService.announceToScreenReader(message)` - Announce message to screen readers
- `ModalService.setAriaLabels(labels)` - Configure ARIA labels for components

Performance

- `ModalService.preload(components)` - Preload modal components for faster opening
- `ModalService.enableVirtualization()` - Enable virtualization for large modal stacks
- `ModalService.setAnimationConfig(config)` - Configure animation performance settings
- `ModalService.measurePerformance()` - Get performance metrics for modal operations
- `ModalService.optimize()` - Apply automatic performance optimizations

Performance and Scalability

☐ Back to Top

Memory Management

☐ Back to Top

Component Lifecycle Optimization ☐ Back to Top

```
graph TD
    A[Modal Mount] --> B[Initialize State<br/>Minimal initial render]
    B --> C[Portal Creation<br/>Create DOM mount point]
    C --> D[Event Listener Setup<br/>Global keyboard, resize]
    D --> E[Animation Setup<br/>CSS class preparation]

    E --> F[Content Render<br/>Main modal content]
    F --> G[Focus Management<br/>Trap activation]
    G --> H[Accessibility Setup<br/>ARIA attributes]

    H --> I[Modal Active State<br/>User interaction ready]

    I --> J[Modal Close Triggered]
    J --> K[Animation Cleanup<br/>Remove CSS classes]
    K --> L[Focus Restoration<br/>Return to previous element]
    L --> M[Event Cleanup<br/>Remove global listeners]
    M --> N[Portal Cleanup<br/>Remove DOM mount point]
    N --> O[Component Unmount<br/>Memory release]

    subgraph "Memory Optimization"
        P[Weak References<br/>Avoid circular refs]
        Q[Event Delegation<br/>Minimize listeners]
        R[Lazy Loading<br/>Content on demand]
        S[Cleanup Timeouts<br/>Clear pending timers]
    end

    D --> P
    F --> Q
    G --> R
    K --> S
```

Performance Monitoring

□ Back to Top

Real-time Performance Metrics □ Back to Top

```
graph TB
    subgraph "Rendering Performance"
        RENDER_TIME[Render Time<br/>Component mount to paint]
        ANIMATION_FPS[Animation FPS<br/>Transition smoothness]
        LAYOUT_SHIFTS[Layout Shifts<br/>CLS measurement]
        PAINT_TIMING[Paint Timing<br/>First/largest contentful paint]
    end

    subgraph "Interaction Performance"
        FOCUS_LATENCY[Focus Latency<br/>Tab navigation response]
        KEYBOARD_RESPONSE[Keyboard Response<br/>Event handler execution]
        TOUCH_RESPONSE[Touch Response<br/>Mobile interaction timing]
        ACCESSIBILITY_TIMING[Accessibility Timing<br/>Screen reader delays]
    end

    subgraph "Memory Metrics"
        COMPONENT_COUNT[Component Count<br/>Active modal instances]
        EVENT_LISTENERS[Event Listeners<br/>Global listener count]
        DOM_NODES[DOM Nodes<br/>Modal-related elements]
        MEMORY_LEAKS[Memory Leaks<br/>Unreleased references]
    end

    subgraph "User Experience"
        MODAL_BOUNCE_RATE[Modal Bounce Rate<br/>Immediate closures]
        COMPLETION_RATE[Completion Rate<br/>Successful actions]
        ERROR_RATE[Error Rate<br/>Failed operations]
        ACCESSIBILITY_SCORE[Accessibility Score<br/>WCAG compliance]
    end

    RENDER_TIME --> MODAL_BOUNCE_RATE
    ANIMATION_FPS --> COMPLETION_RATE
    FOCUS_LATENCY --> ERROR_RATE
    KEYBOARD_RESPONSE --> ACCESSIBILITY_SCORE
```

Security and Privacy

Back to Top

Security Considerations

Back to Top

Content Security and XSS Prevention Back to Top

```
graph TD
    subgraph "Input Sanitization"
        CONTENT_VALIDATION[Content Validation<br/>HTML sanitization]
        XSS_PREVENTION[XSS Prevention<br/>Script tag filtering]
        URL_VALIDATION[URL Validation<br/>Safe link checking]
        FILE_VALIDATION[File Validation<br/>Upload security]
    end

    subgraph "DOM Security"
        PORTAL_ISOLATION[Portal Isolation<br/>Separate render context]
        EVENT_SANITIZATION[Event Sanitization<br/>Handler validation]
        ATTRIBUTE_FILTERING[Attribute Filtering<br/>Dangerous attr removal]
        SCRIPT_ISOLATION[Script Isolation<br/>Execution context control]
    end

    subgraph "Access Control"
        PERMISSION_CHECK[Permission Check<br/>User authorization]
        ROLE_VALIDATION[Role Validation<br/>Feature access control]
        SESSION_VALIDATION[Session Validation<br/>Authentication status]
        RATE_LIMITING[Rate Limiting<br/>Abuse prevention]
    end

    CONTENT_VALIDATION --> PORTAL_ISOLATION
    XSS_PREVENTION --> EVENT_SANITIZATION
    URL_VALIDATION --> ATTRIBUTE_FILTERING
    FILE_VALIDATION --> SCRIPT_ISOLATION

    PORTAL_ISOLATION --> PERMISSION_CHECK
    EVENT_SANITIZATION --> ROLE_VALIDATION
    ATTRIBUTE_FILTERING --> SESSION_VALIDATION
    SCRIPT_ISOLATION --> RATE_LIMITING
```

Testing, Monitoring, and Maintainability

Back to Top

Comprehensive Testing Strategy

Back to Top

Multi-Layer Testing Approach Back to Top

```
graph TD
    subgraph "Unit Tests"
        UT1[Focus Management Tests<br/>Tab navigation logic]
        UT2[Animation Tests<br/>Transition states]
        UT3[State Management Tests<br/>Modal lifecycle]
        UT4[Accessibility Tests<br/>ARIA attributes]
    end

    subgraph "Integration Tests"
        IT1[Component Integration<br/>Parent-child communication]
        IT2[Event Handling<br/>Keyboard, mouse, touch]
        IT3[Portal Rendering<br/>DOM manipulation]
        IT4[Theme Integration<br/>Style application]
    end

    subgraph "E2E Tests"
        E2E1[User Journey Tests<br/>Complete workflows]
        E2E2[Cross-browser Tests<br/>Browser compatibility]
        E2E3[Device Testing<br/>Mobile, tablet, desktop]
        E2E4[Performance Tests<br/>Load time, animation fps]
    end

    subgraph "Accessibility Tests"
        AT1[Screen Reader Tests<br/>NVDA, JAWS, VoiceOver]
        AT2[Keyboard Navigation<br/>Tab order, shortcuts]
        AT3[Color Contrast<br/>WCAG compliance]
        AT4[Focus Management<br/>Visual indicators]
    end
```

```
UT1 --> IT1
UT2 --> IT2
UT3 --> IT3
UT4 --> IT4

IT1 --> E2E1
IT2 --> E2E2
IT3 --> E2E3
IT4 --> E2E4

E2E1 --> AT1
E2E2 --> AT2
E2E3 --> AT3
E2E4 --> AT4
```

Trade-offs, Deep Dives, and Extensions

☐ Back to Top

Modal vs Alternative Patterns

☐ Back to Top

Pattern	Modal Dialog	Slide Panel	Inline Expansion	New Page
Context Preservation	Excellent	Good	Excellent	Poor
Mobile Experience	Good	Excellent	Good	Excellent
Accessibility	Complex	Moderate	Simple	Simple
Performance	Good	Good	Excellent	Variable
SEO Impact	None	None	None	Positive
Deep Linking	Difficult	Difficult	Possible	Natural

Advanced Features

☐ Back to Top

Intelligent Modal Behavior ☐ Back to Top

```
graph TD
    subgraph "Adaptive Behavior"
        DEVICE_DETECTION[Device Detection<br/>Mobile, tablet, desktop]
        USAGE_PATTERNS[Usage Patterns<br/>User behavior analysis]
        CONTEXT_AWARENESS[Context Awareness<br/>Page state, user flow]
        PERFORMANCE_ADAPTATION[Performance Adaptation<br/>Hardware capabilities]
    end

    subgraph "Smart Positioning"
        VIEWPORT_ANALYSIS[Viewport Analysis<br/>Available space calculation]
        COLLISION_DETECTION[Collision Detection<br/>Overlap avoidance]
        SMART_SIZING[Smart Sizing<br/>Content-aware dimensions]
        MULTI_SCREEN[Multi-screen Support<br/>Display awareness]
    end

    subgraph "Predictive Features"
        PRELOAD_CONTENT[Preload Content<br/>Anticipated modals]
        GESTURE_PREDICTION[Gesture Prediction<br/>Touch pattern learning]
        ACCESSIBILITY_PREDICTION[Accessibility Prediction<br/>User needs detection]
        PERFORMANCE_PREDICTION[Performance Prediction<br/>Resource planning]
    end

    DEVICE_DETECTION --> VIEWPORT_ANALYSIS
    USAGE_PATTERNS --> COLLISION_DETECTION
    CONTEXT_AWARENESS --> SMART_SIZING
    PERFORMANCE_ADAPTATION --> MULTI_SCREEN

    VIEWPORT_ANALYSIS --> PRELOAD_CONTENT
    COLLISION_DETECTION --> GESTURE_PREDICTION
    SMART_SIZING --> ACCESSIBILITY_PREDICTION
    MULTI_SCREEN --> PERFORMANCE_PREDICTION
```

Future Extensions

☐ Back to Top

Next-Generation Modal Features ☐ Back to Top

---

1. **Voice Integration**:

- Voice-controlled modal navigation
- Speech-to-text for form inputs
- Audio feedback for actions
- Voice accessibility features

2. **Gesture Recognition**:
   - Touch gesture controls
   - Eye tracking navigation
   - Hand gesture detection
   - Spatial interaction support
3. **AI-Powered UX**:
   - Intent prediction for modal content
   - Adaptive layouts based on usage
   - Personalized interaction patterns
   - Intelligent focus management
4. **Immersive Technologies**:
   - AR/VR modal overlays
   - 3D spatial positioning
   - Haptic feedback integration
   - Immersive interaction paradigms

This comprehensive design provides a robust foundation for building an accessible, performant, and flexible modal dialog system that handles complex focus management, provides excellent user experience across all devices, and maintains high accessibility standards while being extensible for future enhancements.

TypeScript Interfaces & Component Props

☐ Back to Top

Core Data Interfaces

```
interface ModalConfig {
  id: string;
  type: 'alert' | 'confirm' | 'prompt' | 'custom';
  title?: string;
  content: React.ReactNode | string;
  size: 'sm' | 'md' | 'lg' | 'xl' | 'fullscreen';
  position: 'center' | 'top' | 'bottom' | 'custom';
  backdrop: boolean | 'static';
  keyboard: boolean;
  animation: 'fade' | 'slide' | 'zoom' | 'none';
  zIndex?: number;
}
```

```typescript
interface ModalAction {
  id: string;
  label: string;
  variant: 'primary' | 'secondary' | 'danger' | 'success';
  onClick: (modal: ModalInstance) => void | Promise<void>;
  disabled?: boolean;
  loading?: boolean;
  autoClose?: boolean;
}

interface ModalInstance {
  id: string;
  config: ModalConfig;
  isOpen: boolean;
  isAnimating: boolean;
  openedAt: Date;
  data?: any;
  resolve?: (value: any) => void;
  reject?: (reason: any) => void;
}

interface ModalState {
  instances: ModalInstance[];
  activeModalId?: string;
  maxZIndex: number;
  focusHistory: HTMLElement[];
  bodyScrollY: number;
  isBodyScrollLocked: boolean;
}

interface DialogResult<T = any> {
  confirmed: boolean;
  data?: T;
  action?: string;
}
```

Component Props Interfaces

```typescript
interface ModalProps {
  isOpen: boolean;
  onClose: () => void;
  title?: string;
  size?: ModalSize;
  backdrop?: boolean | 'static';
  keyboard?: boolean;
```

```
  animation?: AnimationType;
  className?: string;
  style?: React.CSSProperties;
  children: React.ReactNode;
  zIndex?: number;
}

interface ModalHeaderProps {
  title?: string;
  showCloseButton?: boolean;
  onClose?: () => void;
  className?: string;
  children?: React.ReactNode;
}

interface ModalBodyProps {
  className?: string;
  scrollable?: boolean;
  maxHeight?: string;
  children: React.ReactNode;
}

interface ModalFooterProps {
  actions?: ModalAction[];
  align?: 'left' | 'center' | 'right' | 'space-between';
  className?: string;
  children?: React.ReactNode;
}

interface ConfirmDialogProps {
  title?: string;
  message: string;
  confirmText?: string;
  cancelText?: string;
  variant?: 'default' | 'danger';
  onConfirm?: () => void | Promise<void>;
  onCancel?: () => void;
}

interface PromptDialogProps {
  title?: string;
  message: string;
  placeholder?: string;
  defaultValue?: string;
  validator?: (value: string) => string | null;
```

```
  onSubmit?: (value: string) => void | Promise<void>;
  onCancel?: () => void;
}
```

API Reference

☐ Back to Top

Modal Management

- `ModalService.open(config)` - Open new modal with configuration options
- `ModalService.close(modalId?)` - Close specific modal or topmost modal
- `ModalService.closeAll()` - Close all open modals in stack
- `ModalService.update(modalId, config)` - Update modal configuration dynamically
- `ModalService.isOpen(modalId?)` - Check if modal is open by ID or any modal

Dialog Utilities

- `ModalService.alert(message, options?)` - Show alert dialog with OK button
- `ModalService.confirm(message, options?)` - Show confirmation dialog with Yes/No
- `ModalService.prompt(message, options?)` - Show input prompt dialog
- `ModalService.custom(component, props?)` - Open custom modal component
- `ModalService.loading(message?, options?)` - Show loading modal with spinner

Stack Management

- `ModalService.getStack()` - Get current modal stack information
- `ModalService.setMaxModals(limit)` - Set maximum concurrent modals allowed
- `ModalService.bringToFront(modalId)` - Bring specific modal to front of stack
- `ModalService.sendToBack(modalId)` - Send modal to back of stack
- `ModalService.getActiveModal()` - Get currently active (topmost) modal

Event Management

- `ModalService.on(event, callback)` - Subscribe to modal events
- `ModalService.off(event, callback)` - Unsubscribe from modal events
- `ModalService.emit(event, data)` - Emit custom modal event
- Events: `'open' | 'close' | 'beforeClose' | 'afterOpen' | 'stackChange'`

Configuration

- `ModalService.setDefaults(config)` - Set default modal configuration
- `ModalService.getDefaults()` - Get current default configuration
- `ModalService.registerAnimation(name, config)` - Register custom animation
- `ModalService.setTheme(theme)` - Apply theme to all modals
- `ModalService.resetConfig()` - Reset to factory default configuration

Accessibility

- `ModalService.setA11yConfig(config)` - Configure accessibility options
- `ModalService.enableFocusTrap(modalId)` - Enable focus trapping for modal
- `ModalService.disableFocusTrap(modalId)` - Disable focus trapping
- `ModalService.announceToScreenReader(message)` - Announce message to screen readers
- `ModalService.setAriaLabels(labels)` - Configure ARIA labels for components

Performance

- `ModalService.preload(components)` - Preload modal components for faster opening
- `ModalService.enableVirtualization()` - Enable virtualization for large modal stacks
- `ModalService.setAnimationConfig(config)` - Configure animation performance settings
- `ModalService.measurePerformance()` - Get performance metrics for modal operations
- `ModalService.optimize()` - Apply automatic performance optimizations

Performance and Scalability

☐ Back to Top

Memory Management

☐ Back to Top

Component Lifecycle Optimization ☐ Back to Top

```
graph TD
    A[Modal Mount] --> B[Initialize State<br/>Minimal initial render]
    B --> C[Portal Creation<br/>Create DOM mount point]
    C --> D[Event Listener Setup<br/>Global keyboard, resize]
    D --> E[Animation Setup<br/>CSS class preparation]

    E --> F[Content Render<br/>Main modal content]
    F --> G[Focus Management<br/>Trap activation]
    G --> H[Accessibility Setup<br/>ARIA attributes]

    H --> I[Modal Active State<br/>User interaction ready]

    I --> J[Modal Close Triggered]
    J --> K[Animation Cleanup<br/>Remove CSS classes]
    K --> L[Focus Restoration<br/>Return to previous element]
    L --> M[Event Cleanup<br/>Remove global listeners]
    M --> N[Portal Cleanup<br/>Remove DOM mount point]
    N --> O[Component Unmount<br/>Memory release]

    subgraph "Memory Optimization"
        P[Weak References<br/>Avoid circular refs]
        Q[Event Delegation<br/>Minimize listeners]
        R[Lazy Loading<br/>Content on demand]
        S[Cleanup Timeouts<br/>Clear pending timers]
    end

    D --> P
    F --> Q
    G --> R
    K --> S
```

Performance Monitoring

□ Back to Top

Real-time Performance Metrics □ Back to Top

```
graph TB
    subgraph "Rendering Performance"
        RENDER_TIME[Render Time<br/>Component mount to paint]
        ANIMATION_FPS[Animation FPS<br/>Transition smoothness]
        LAYOUT_SHIFTS[Layout Shifts<br/>CLS measurement]
```

```
    PAINT_TIMING[Paint Timing<br/>First/largest contentful paint]
end

subgraph "Interaction Performance"
    FOCUS_LATENCY[Focus Latency<br/>Tab navigation response]
    KEYBOARD_RESPONSE[Keyboard Response<br/>Event handler execution]
    TOUCH_RESPONSE[Touch Response<br/>Mobile interaction timing]
    ACCESSIBILITY_TIMING[Accessibility Timing<br/>Screen reader delays]
end

subgraph "Memory Metrics"
    COMPONENT_COUNT[Component Count<br/>Active modal instances]
    EVENT_LISTENERS[Event Listeners<br/>Global listener count]
    DOM_NODES[DOM Nodes<br/>Modal-related elements]
    MEMORY_LEAKS[Memory Leaks<br/>Unreleased references]
end

subgraph "User Experience"
    MODAL_BOUNCE_RATE[Modal Bounce Rate<br/>Immediate closures]
    COMPLETION_RATE[Completion Rate<br/>Successful actions]
    ERROR_RATE[Error Rate<br/>Failed operations]
    ACCESSIBILITY_SCORE[Accessibility Score<br/>WCAG compliance]
end

RENDER_TIME --> MODAL_BOUNCE_RATE
ANIMATION_FPS --> COMPLETION_RATE
FOCUS_LATENCY --> ERROR_RATE
KEYBOARD_RESPONSE --> ACCESSIBILITY_SCORE
```

Security and Privacy

☐ Back to Top

Security Considerations

☐ Back to Top

Content Security and XSS Prevention ☐ Back to Top

```
graph TD
    subgraph "Input Sanitization"
        CONTENT_VALIDATION[Content Validation<br/>HTML sanitization]
        XSS_PREVENTION[XSS Prevention<br/>Script tag filtering]
        URL_VALIDATION[URL Validation<br/>Safe link checking]
        FILE_VALIDATION[File Validation<br/>Upload security]
    end

    subgraph "DOM Security"
        PORTAL_ISOLATION[Portal Isolation<br/>Separate render context]
        EVENT_SANITIZATION[Event Sanitization<br/>Handler validation]
        ATTRIBUTE_FILTERING[Attribute Filtering<br/>Dangerous attr removal]
        SCRIPT_ISOLATION[Script Isolation<br/>Execution context control]
    end

    subgraph "Access Control"
        PERMISSION_CHECK[Permission Check<br/>User authorization]
        ROLE_VALIDATION[Role Validation<br/>Feature access control]
        SESSION_VALIDATION[Session Validation<br/>Authentication status]
        RATE_LIMITING[Rate Limiting<br/>Abuse prevention]
    end

    CONTENT_VALIDATION --> PORTAL_ISOLATION
    XSS_PREVENTION --> EVENT_SANITIZATION
    URL_VALIDATION --> ATTRIBUTE_FILTERING
    FILE_VALIDATION --> SCRIPT_ISOLATION

    PORTAL_ISOLATION --> PERMISSION_CHECK
    EVENT_SANITIZATION --> ROLE_VALIDATION
    ATTRIBUTE_FILTERING --> SESSION_VALIDATION
    SCRIPT_ISOLATION --> RATE_LIMITING
```

Testing, Monitoring, and Maintainability

☐ Back to Top

Comprehensive Testing Strategy

☐ Back to Top

Multi-Layer Testing Approach □ Back to Top

```
graph TD
    subgraph "Unit Tests"
        UT1[Focus Management Tests<br/>Tab navigation logic]
        UT2[Animation Tests<br/>Transition states]
        UT3[State Management Tests<br/>Modal lifecycle]
        UT4[Accessibility Tests<br/>ARIA attributes]
    end

    subgraph "Integration Tests"
        IT1[Component Integration<br/>Parent-child communication]
        IT2[Event Handling<br/>Keyboard, mouse, touch]
        IT3[Portal Rendering<br/>DOM manipulation]
        IT4[Theme Integration<br/>Style application]
    end

    subgraph "E2E Tests"
        E2E1[User Journey Tests<br/>Complete workflows]
        E2E2[Cross-browser Tests<br/>Browser compatibility]
        E2E3[Device Testing<br/>Mobile, tablet, desktop]
        E2E4[Performance Tests<br/>Load time, animation fps]
    end

    subgraph "Accessibility Tests"
        AT1[Screen Reader Tests<br/>NVDA, JAWS, VoiceOver]
        AT2[Keyboard Navigation<br/>Tab order, shortcuts]
        AT3[Color Contrast<br/>WCAG compliance]
        AT4[Focus Management<br/>Visual indicators]
    end

    UT1 --> IT1
    UT2 --> IT2
    UT3 --> IT3
    UT4 --> IT4

    IT1 --> E2E1
    IT2 --> E2E2
    IT3 --> E2E3
    IT4 --> E2E4

    E2E1 --> AT1
    E2E2 --> AT2
    E2E3 --> AT3
```

```
    E2E4 --> AT4
```

Trade-offs, Deep Dives, and Extensions

☐ Back to Top

Modal vs Alternative Patterns

☐ Back to Top

Pattern	Modal Dialog	Slide Panel	Inline Expansion	New Page
Context Preservation	Excellent	Good	Excellent	Poor
Mobile Experience	Good	Excellent	Good	Excellent
Accessibility	Complex	Moderate	Simple	Simple
Performance	Good	Good	Excellent	Variable
SEO Impact	None	None	None	Positive
Deep Linking	Difficult	Difficult	Possible	Natural

Advanced Features

☐ Back to Top

Intelligent Modal Behavior ☐ Back to Top

```
graph TD
    subgraph "Adaptive Behavior"
        DEVICE_DETECTION[Device Detection<br/>Mobile, tablet, desktop]
        USAGE_PATTERNS[Usage Patterns<br/>User behavior analysis]
        CONTEXT_AWARENESS[Context Awareness<br/>Page state, user flow]
        PERFORMANCE_ADAPTATION[Performance Adaptation<br/>Hardware capabilities]
    end

    subgraph "Smart Positioning"
        VIEWPORT_ANALYSIS[Viewport Analysis<br/>Available space calculation]
        COLLISION_DETECTION[Collision Detection<br/>Overlap avoidance]
        SMART_SIZING[Smart Sizing<br/>Content-aware dimensions]
```

```
    MULTI_SCREEN[Multi-screen Support<br/>Display awareness]
end

subgraph "Predictive Features"
    PRELOAD_CONTENT[Preload Content<br/>Anticipated modals]
    GESTURE_PREDICTION[Gesture Prediction<br/>Touch pattern learning]
    ACCESSIBILITY_PREDICTION[Accessibility Prediction<br/>User needs detection]
    PERFORMANCE_PREDICTION[Performance Prediction<br/>Resource planning]
end

DEVICE_DETECTION --> VIEWPORT_ANALYSIS
USAGE_PATTERNS --> COLLISION_DETECTION
CONTEXT_AWARENESS --> SMART_SIZING
PERFORMANCE_ADAPTATION --> MULTI_SCREEN

VIEWPORT_ANALYSIS --> PRELOAD_CONTENT
COLLISION_DETECTION --> GESTURE_PREDICTION
SMART_SIZING --> ACCESSIBILITY_PREDICTION
MULTI_SCREEN --> PERFORMANCE_PREDICTION
```

Future Extensions

☐ Back to Top

Next-Generation Modal Features ☐ Back to Top

---

1. **Voice Integration**:
   - Voice-controlled modal navigation
   - Speech-to-text for form inputs
   - Audio feedback for actions
   - Voice accessibility features
2. **Gesture Recognition**:
   - Touch gesture controls
   - Eye tracking navigation
   - Hand gesture detection
   - Spatial interaction support
3. **AI-Powered UX**:
   - Intent prediction for modal content
   - Adaptive layouts based on usage
   - Personalized interaction patterns
   - Intelligent focus management
4. **Immersive Technologies**:

- AR/VR modal overlays
- 3D spatial positioning
- Haptic feedback integration
- Immersive interaction paradigms

This comprehensive design provides a robust foundation for building an accessible, performant, and flexible modal dialog system that handles complex focus management, provides excellent user experience across all devices, and maintains high accessibility standards while being extensible for future enhancements.