IndoSwap Angular 17+ - Modern Copy-Paste Guide with Signals

🚀 Terminal Setup Commands

bash

```bash
# 1. Install latest Angular CLI
npm install -g @angular/cli@17

# 2. Create project with standalone components
ng new indoswap-angular --style=scss --routing=false --standalone=true
cd indoswap-angular

# 3. Generate standalone components
ng g c components/header --standalone=true --skip-tests=true
ng g c components/hero --standalone=true --skip-tests=true
ng g c components/stats --standalone=true --skip-tests=true
ng g c components/swap --standalone=true --skip-tests=true
ng g c components/features --standalone=true --skip-tests=true
ng g c components/footer --standalone=true --skip-tests=true
ng g c components/notification --standalone=true --skip-tests=true

# 4. Generate services
ng g s services/wallet --skip-tests=true
ng g s services/swap --skip-tests=true
ng g s services/notification --skip-tests=true

# 5. Create models folder
mkdir src/app/models
```

📁 File 1: Modern Token Interface with Type Safety

File: `src/app/models/token.interface.ts` **Action:** Create this file and copy:

typescript

```typescript
export interface Token {
  readonly symbol: string;
  readonly name: string;
  readonly icon: string;
  readonly balance: number;
  readonly address?: string;
  readonly decimals?: number;
}

export interface SwapData {
  readonly fromToken: Token;
  readonly toToken: Token;
  readonly fromAmount: number;
  readonly toAmount: number;
  readonly exchangeRate: number;
  readonly priceImpact: number;
  readonly tradingFee: number;
  readonly minimumReceived: number;
  readonly slippage: number;
}

export interface NotificationData {
  readonly message: string;
  readonly type: 'success' | 'error' | 'info' | 'warning';
  readonly duration?: number;
  readonly id?: string;
}

export type WalletStatus = 'disconnected' | 'connecting' | 'connected' | 'error';

export interface WalletState {
  readonly status: WalletStatus;
  readonly address: string;
  readonly chainId?: number;
  readonly balance?: string;
}
```

🔧 File 2: Modern Wallet Service with Signals

File: `src/app/services/wallet.service.ts` **Action:** Replace entire content with:

typescript

```typescript
import { Injectable, signal, computed } from '@angular/core';
import { WalletState, WalletStatus } from '../models/token.interface';

@Injectable({
  providedIn: 'root'
})
export class WalletService {
  // Signals for reactive state management
  private readonly walletState = signal<WalletState>({
    status: 'disconnected',
    address: '',
    chainId: 56, // BSC mainnet
    balance: '0'
  });

  // Public computed signals
  readonly isConnected = computed(() => this.walletState().status === 'connected');
  readonly isConnecting = computed(() => this.walletState().status === 'connecting');
  readonly address = computed(() => this.walletState().address);
  readonly shortAddress = computed(() => {
    const addr = this.address();
    return addr ? `${addr.slice(0, 6)}...${addr.slice(-4)}` : '';
  });
  readonly chainId = computed(() => this.walletState().chainId);
  readonly balance = computed(() => this.walletState().balance);

  async connectWallet(): Promise<void> {
    this.walletState.update(state => ({
      ...state,
      status: 'connecting'
    }));

    try {
      // Simulate wallet connection
      await new Promise(resolve => setTimeout(resolve, 1500));

      // Mock successful connection
      this.walletState.set({
        status: 'connected',
        address: '0x742e4B537583D5cB93f7FD5B23982F1dA5096e7F',
        chainId: 56,
        balance: '2.45'
      });
    } catch (error) {
      this.walletState.update(state => ({
        ...state,
```

```
        status: 'error'
      }));
      throw error;
    }
  }

  disconnectWallet(): void {
    this.walletState.set({
      status: 'disconnected',
      address: '',
      chainId: 56,
      balance: '0'
    });
  }

  // For reactive updates from external wallet events
  updateWalletState(updates: Partial<WalletState>): void {
    this.walletState.update(state => ({ ...state, ...updates }));
  }
}
```

🔄 File 3: Modern Swap Service with Signals

File: `src/app/services/swap.service.ts` **Action:** Replace entire content with:

typescript

```typescript
import { Injectable, signal, computed, effect } from '@angular/core';
import { Token, SwapData } from '../models/token.interface';

@Injectable({
  providedIn: 'root'
})
export class SwapService {
  private readonly exchangeRates = new Map([
    ['BNB-BUSD', 285.43],
    ['BUSD-BNB', 0.0035],
    ['BNB-USDT', 284.87],
    ['USDT-BNB', 0.0035],
    ['BUSD-USDT', 0.998],
    ['USDT-BUSD', 1.002]
  ]);

  private readonly defaultTokens: readonly Token[] = [
    { symbol: 'BNB', name: 'Binance Coin', icon: 'bnb', balance: 2.45, decimals: 18 },
    { symbol: 'BUSD', name: 'Binance USD', icon: 'busd', balance: 1250.00, decimals: 18 },
    { symbol: 'USDT', name: 'Tether USD', icon: 'usdt', balance: 500.00, decimals: 6 }
  ] as const;

  // Signals for state management
  private readonly fromToken = signal<Token>(this.defaultTokens[0]);
  private readonly toToken = signal<Token>(this.defaultTokens[1]);
  private readonly fromAmount = signal<number>(0);
  private readonly slippage = signal<number>(0.5);
  private readonly isSwapping = signal<boolean>(false);

  // Computed values
  readonly currentFromToken = computed(() => this.fromToken());
  readonly currentToToken = computed(() => this.toToken());
  readonly currentFromAmount = computed(() => this.fromAmount());
  readonly currentSlippage = computed(() => this.slippage());
  readonly isCurrentlySwapping = computed(() => this.isSwapping());

  readonly exchangeRate = computed(() => {
    const from = this.fromToken().symbol;
    const to = this.toToken().symbol;
    const rateKey = `${from}-${to}`;
    return this.exchangeRates.get(rateKey) ?? 1;
  });

  readonly toAmount = computed(() => {
    const amount = this.fromAmount();
    const rate = this.exchangeRate();
```

```
    return amount * rate;
  });

  readonly minimumReceived = computed(() => {
    const amount = this.toAmount();
    const slippagePercent = this.slippage() / 100;
    return amount * (1 - slippagePercent);
  });

  readonly priceImpact = computed(() => {
    const amount = this.fromAmount();
    // Simulate price impact based on amount
    if (amount === 0) return 0;
    return Math.min(amount * 0.001, 5); // Max 5% impact
  });

  readonly swapData = computed<SwapData>(() => ({
    fromToken: this.fromToken(),
    toToken: this.toToken(),
    fromAmount: this.fromAmount(),
    toAmount: this.toAmount(),
    exchangeRate: this.exchangeRate(),
    priceImpact: this.priceImpact(),
    tradingFee: 0.25,
    minimumReceived: this.minimumReceived(),
    slippage: this.slippage()
  }));

  // Effect for logging swap data changes (optional)
  private swapDataEffect = effect(() => {
    const data = this.swapData();
    if (data.fromAmount > 0) {
      console.log('Swap data updated:', data);
    }
  });

  // Public methods
  getAvailableTokens(): readonly Token[] {
    return this.defaultTokens;
  }

  updateFromAmount(amount: number): void {
    this.fromAmount.set(Math.max(0, amount));
  }

  updateSlippage(slippage: number): void {
    this.slippage.set(Math.max(0.1, Math.min(50, slippage)));
```

```
}

swapTokens(): void {
  const currentFrom = this.fromToken();
  const currentTo = this.toToken();

  this.fromToken.set(currentTo);
  this.toToken.set(currentFrom);
  this.fromAmount.set(0);
}

selectFromToken(token: Token): void {
  if (token.symbol !== this.toToken().symbol) {
    this.fromToken.set(token);
  }
}

selectToToken(token: Token): void {
  if (token.symbol !== this.fromToken().symbol) {
    this.toToken.set(token);
  }
}

async executeSwap(): Promise<boolean> {
  this.isSwapping.set(true);

  try {
    // Simulate blockchain transaction
    await new Promise(resolve => setTimeout(resolve, 2000));

    // Reset form on success
    this.fromAmount.set(0);
    return true;
  } catch (error) {
    throw error;
  } finally {
    this.isSwapping.set(false);
  }
}

// For real-time price updates
updateExchangeRate(fromSymbol: string, toSymbol: string, rate: number): void {
  const key = `${fromSymbol}-${toSymbol}`;
  this.exchangeRates.set(key, rate);
  // Trigger recalculation by updating a signal that computed values depend on
  this.fromAmount.update(amount => amount);
```

```
    }
  }
```

🔔 File 4: Modern Notification Service with Signals

File: `src/app/services/notification.service.ts` **Action:** Replace entire content with:

typescript

```typescript
import { Injectable, signal, computed } from '@angular/core';
import { NotificationData } from '../models/token.interface';

@Injectable({
  providedIn: 'root'
})
export class NotificationService {
  private readonly notifications = signal<NotificationData[]>([]);
  private notificationIdCounter = 0;

  // Public computed signals
  readonly currentNotifications = computed(() => this.notifications());
  readonly hasNotifications = computed(() => this.notifications().length > 0);
  readonly latestNotification = computed(() => {
    const notifications = this.notifications();
    return notifications[notifications.length - 1] || null;
  });

  showNotification(
    message: string,
    type: NotificationData['type'] = 'info',
    duration: number = 3000
  ): string {
    const id = `notification-${++this.notificationIdCounter}`;
    const notification: NotificationData = {
      id,
      message,
      type,
      duration
    };

    // Add notification
    this.notifications.update(notifications => [...notifications, notification]);

    // Auto-remove after duration
    if (duration > 0) {
      setTimeout(() => {
        this.removeNotification(id);
      }, duration);
    }

    return id;
  }

  removeNotification(id: string): void {
    this.notifications.update(notifications =>
```

```
      notifications.filter(n => n.id !== id)
    );
  }

  clearAllNotifications(): void {
    this.notifications.set([]);
  }

  // Convenience methods
  showSuccess(message: string, duration?: number): string {
    return this.showNotification(message, 'success', duration);
  }

  showError(message: string, duration?: number): string {
    return this.showNotification(message, 'error', duration);
  }

  showWarning(message: string, duration?: number): string {
    return this.showNotification(message, 'warning', duration);
  }

  showInfo(message: string, duration?: number): string {
    return this.showNotification(message, 'info', duration);
  }
}
```

🏠 File 5: Modern App Component (Standalone)

File: `src/app/app.component.ts` **Action:** Replace entire content with:

typescript

```typescript
import { Component } from '@angular/core';
import { CommonModule } from '@angular/common';
import { HeaderComponent } from './components/header/header.component';
import { HeroComponent } from './components/hero/hero.component';
import { StatsComponent } from './components/stats/stats.component';
import { SwapComponent } from './components/swap/swap.component';
import { FeaturesComponent } from './components/features/features.component';
import { FooterComponent } from './components/footer/footer.component';
import { NotificationComponent } from './components/notification/notification.component';

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [
    CommonModule,
    HeaderComponent,
    HeroComponent,
    StatsComponent,
    SwapComponent,
    FeaturesComponent,
    FooterComponent,
    NotificationComponent
  ],
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})
export class AppComponent {
  readonly title = 'IndoSwap - Next Evolution DeFi Exchange';
}
```

File: `src/app/app.component.html` **Action:** Replace entire content with:

```html
<div class="background-animation">
  <div class="bg-circle"></div>
  <div class="bg-circle"></div>
  <div class="bg-circle"></div>
</div>

<app-header />

<div class="container">
  <app-hero />
  <app-stats />
  <app-swap />
  <app-features />
</div>

<app-footer />
<app-notification />
```

File: `src/app/app.component.scss` **Action:** Create/replace with:

scss

```scss
:host {
  display: block;
  min-height: 100vh;
}

.background-animation {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  z-index: -1;
  opacity: 0.1;
  pointer-events: none;
}

.bg-circle {
  position: absolute;
  border-radius: 50%;
  background: rgba(255, 255, 255, 0.1);
  animation: float 6s ease-in-out infinite;

  &:nth-child(1) {
    width: 80px;
    height: 80px;
    top: 20%;
    left: 10%;
    animation-delay: -2s;
  }

  &:nth-child(2) {
    width: 120px;
    height: 120px;
    top: 60%;
    right: 10%;
    animation-delay: -4s;
  }

  &:nth-child(3) {
    width: 60px;
    height: 60px;
    top: 80%;
    left: 80%;
    animation-delay: -1s;
  }
}
```

```scss
@keyframes float {
  0%, 100% {
    transform: translateY(0px) rotate(0deg);
  }
  50% {
    transform: translateY(-20px) rotate(180deg);
  }
}

.container {
  max-width: 1200px;
  margin: 0 auto;
  padding: 2rem;

  @media (max-width: 768px) {
    padding: 1rem;
  }
}
```

📦 File 6: Modern Bootstrap (main.ts)

File: `src/main.ts` **Action:** Replace entire content with:

```typescript
import { bootstrapApplication } from '@angular/platform-browser';
import { provideExperimentalZonelessChangeDetection } from '@angular/core';
import { AppComponent } from './app/app.component';

bootstrapApplication(AppComponent, {
  providers: [
    // Enable zoneless change detection for better performance
    provideExperimentalZonelessChangeDetection(),
    // Add other providers here as needed
  ]
}).catch(err => console.error(err));
```

🌐 File 7: Modern Global Styles

File: `src/styles.scss` **Action:** Replace entire content with:

scss

```css
// CSS Reset
*,
*::before,
*::after {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}

// CSS Custom Properties
:root {
  --primary-gradient: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  --glass-bg: rgba(255, 255, 255, 0.1);
  --glass-border: rgba(255, 255, 255, 0.2);
  --text-primary: #ffffff;
  --text-secondary: rgba(255, 255, 255, 0.8);
  --shadow-primary: 0 20px 40px rgba(0, 0, 0, 0.1);
  --border-radius: 16px;
  --transition: all 0.3s cubic-bezier(0.4, 0, 0.2, 1);
}

// Base styles
html {
  font-size: 16px;
  scroll-behavior: smooth;
}

body {
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto,
               'Helvetica Neue', Arial, sans-serif;
  background: var(--primary-gradient);
  min-height: 100vh;
  color: var(--text-primary);
  overflow-x: hidden;
  line-height: 1.6;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

// Focus styles for accessibility
:focus-visible {
  outline: 2px solid #ffd700;
  outline-offset: 2px;
  border-radius: 4px;
}
```

```css
// Utility classes
.sr-only {
  position: absolute;
  width: 1px;
  height: 1px;
  padding: 0;
  margin: -1px;
  overflow: hidden;
  clip: rect(0, 0, 0, 0);
  white-space: nowrap;
  border: 0;
}

// High contrast mode support
@media (prefers-contrast: high) {
  :root {
    --glass-bg: rgba(255, 255, 255, 0.2);
    --glass-border: rgba(255, 255, 255, 0.4);
  }
}

// Reduced motion support
@media (prefers-reduced-motion: reduce) {
  *,
  *::before,
  *::after {
    animation-duration: 0.01ms !important;
    animation-iteration-count: 1 !important;
    transition-duration: 0.01ms !important;
  }
}
```

⚠️ File 8: Update Package.json

File: `package.json` **Action:** Update dependencies section with:

json

```json
{
  "name": "indoswap-angular",
  "version": "1.0.0",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build",
    "watch": "ng build --watch --configuration development",
    "test": "ng test",
    "lint": "ng lint"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "^17.0.0",
    "@angular/common": "^17.0.0",
    "@angular/compiler": "^17.0.0",
    "@angular/core": "^17.0.0",
    "@angular/forms": "^17.0.0",
    "@angular/platform-browser": "^17.0.0",
    "@angular/platform-browser-dynamic": "^17.0.0",
    "rxjs": "~7.8.0",
    "tslib": "^2.3.0",
    "zone.js": "~0.14.0"
  },
  "devDependencies": {
    "@angular-devkit/build-angular": "^17.0.0",
    "@angular/cli": "^17.0.0",
    "@angular/compiler-cli": "^17.0.0",
    "@angular/eslint": "^17.0.0",
    "@typescript-eslint/eslint-plugin": "^6.0.0",
    "@typescript-eslint/parser": "^6.0.0",
    "eslint": "^8.50.0",
    "typescript": "~5.2.0"
  }
}
```

✋ STOP HERE AND TEST

Test the modern setup:

bash

```bash
npm install
ng serve
```

🎯 What's New & Modern:

✅ **Angular 17+ features:**

- Standalone components (no NgModule needed)
- Signals for reactive state management
- Zoneless change detection for better performance
- Modern bootstrapping

✅ **Best practices:**

- `computed()` for derived state
- `effect()` for side effects
- `signal()` for mutable state
- Readonly interfaces
- CSS custom properties
- Accessibility improvements
- Type safety with `as const`

✅ **Performance improvements:**

- Experimental zoneless change detection
- OnPush change detection ready
- Modern CSS with custom properties
- Reduced motion support

Ready for **Part 2 with modern components**? Let me know when Part 1 works!