coForward Chart2

DEVELOPER GUIDE

coForward Chart2 개발자 매뉴얼

설치부터 데이터 연동, 드로잉과 가격 알림, 상태 저장과 템플릿 커스터마이징, 전체 API 레퍼런스까지 담은 통합 가이드입니다.

버전	발행	구성
v0.1.0	coForward	12개 장

목차

01	빠른 시작	최소 예제 · 체인 · 실시간 틱
02	설치 및 빌드	개발 서버 · 배포 빌드
03	기본 설정 (Config)	레이아웃 · fieldMap · 지표
04	데이터 형식 및 로드	setData · addData · needData
05	차트 마크 & 오버레이	SMA · EMA · VWAP · 일목
06	보조지표 (Sub Panel)	RSI · MACD · 10종
07	드로잉 도구	15종 · 스타일 · 가격 알림
08	이벤트 시스템	6개 이벤트 · 십자선
09	상태 저장 & 복원	localStorage · JSON
10	패널 동기화 (PanelSync)	범위 · 십자선 연동
11	HTML 템플릿 커스터마이징	data-coforward-* 속성
12	전체 API 레퍼런스	메서드 · 이벤트 · PanelSync

01

빠른 시작

최소 예제

HTML

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="/dist/template/coForward_chart2.css">
</head>
<body>
  <!-- 차트가 마운트될 컨테이너 (id 필수) -->
  <div id="myChart" style="width:100%;height:500px;"></div>

  <script type="module">
    import { coForward_chart } from '/dist/coForward_chart2.min.js';

    const panel = new coForward_chart()
      .add('myChart')
      .setData(ohlcvArray);
  </script>
</body>
</html>
```

`ohlcvArray` 는 `{ Date, Open, High, Low, Close, Volume }` 형식의 배열입니다. 필드명이 다를 경우 `fieldMap` 옵션으로 매핑하세요. **4. 데이터**

체인 방식

대부분의 메서드는 `this` 를 반환하므로 메서드를 체이닝할 수 있습니다.

JavaScript

```
const panel = new coForward_chart({ displayCnt: 100 })
  .add('myChart')
  .setSymbol({ caption: 'SAMSUNG', symbol: '005930', decimal: 0, currency: 'KRW' })
  .setData(dailyData)
  .addSubPanel({ indicator: 'rsi', period: 14 })
  .addSubPanel({ indicator: 'macd' });
```

분봉 + 실시간 틱

JavaScript

```
// 초기 분봉 데이터 로드
const panel = new coForward_chart()
  .add('myChart')
  .setData(intradayData);

// 틱 수신 시 addData()로 갱신
websocket.onmessage = (e) => {
  const tick = JSON.parse(e.data);
  panel.addData(tick); // 같은 분 → OHLCV 갱신, 새 분 → 봉 추가
};
```

빌드 파일 위치

파일	설명
`dist/coForward_chart2.js`	원본 ESM 번들 (개발/디버깅용)
`dist/coForward_chart2.min.js`	압축 번들 (배포용, ≈410KB)
`dist/template/`	HTML 오버레이 템플릿 (CSS 포함)

02

설치 및 빌드

요구 사항

- Node.js 18+
- npm 9+

의존성 설치

프로젝트 루트에서 실행합니다.

```
Terminal
npm install
```

개발 서버 (Vite)

```
Terminal
npm run dev
```

브라우저에서 `http://localhost:5173/chart2.html`로 접속합니다.

배포 빌드

```
Terminal
npm run build
```

빌드 결과물은 `dist/` 폴더에 생성됩니다.

파일	크기	용도
`dist/coForward_chart2.js`	≈581KB	원본 (소스맵 디버깅)
`dist/coForward_chart2.min.js`	≈410KB	배포용 압축 번들

파일	크기	용도
`dist/template/`	—	HTML 오버레이 템플릿 일체

! 빌드 후 `dist/template/` 폴더를 서버의 `template/` 경로에 배포하세요. 기본 템플릿 경로는 `./template/*.html` 입니다. **11. 템플릿**

소스 직접 사용 (Vite dev 환경)

JavaScript

```
import { coForward_chart, PanelSync } from '/src/coForward_chart2.js';
```

배포 번들 사용 (프로덕션)

HTML

```
<link rel="stylesheet" href="/dist/template/coForward_chart2.css">
<script type="module">
  import { coForward_chart, PanelSync } from '/dist/coForward_chart2.min.js';
</script>
```

03

기본 설정 (Config)

`new coForward_chart(config)` 생성자에 설정 객체를 넘겨 동작을 커스터마이징합니다. 모든 키는 선택 사항이며, 명시하지 않으면 기본값이 사용됩니다.

일반 설정

키	기본값	설명
`displayCnt`	`50`	초기 화면에 표시할 봉 개수
`xAxisMode`	`'session'`	x축 모드. `'session'` (갭 없음) | `'calendar'` (날짜 기준)
`yAxisSide`	`'right'`	y축 위치. `'right'` | `'left'`
`xTickMax`	`10`	x축 눈금 최대 개수
`bgColor`	CSS 변수 `--chart-bg`	차트 배경색
`autoSave`	`true`	드로잉·상태 자동 저장 여부
`autoSaveKeyPrefix`	`'coForward_chart2'`	localStorage 키 접두사
`marks`	`['candle']`	초기 활성화할 마크 목록

레이아웃 (layout)

```JavaScript
const config = {
  layout: {
    main: {
      top: 10, right: 100, bottomWithX: 50, bottomNoX: 0, left: 10,
    },
    sub: {
      top: 0, right: 100, left: 10, bottomWithX: 30, bottomNoX: 0,
    },
    subPanel: {
      height: 100, // 서브패널 기본 높이(px)
      minHeight: 60,
      maxHeight: 300,
    },
    xPadRatio: 0.3,        // x 도메인 여백 비율
    yBottomVolRatio: 0.25, // 볼륨 공간 비율
    yTopPadRatio: 0.05,    // y 상단 여유 비율
    volDisplayRatio: 0.2,  // 볼륨 바 높이 비율
  },
};
```

필드 매핑 (fieldMap)

입력 데이터의 키가 다를 때 내부 표준 키로 매핑합니다.

```JavaScript
const config = {
  fieldMap: {
    date: 'time',  // 기본: 'Date'
    open: 'o',     // 기본: 'Open'
    high: 'h',     // 기본: 'High'
    low: 'l',      // 기본: 'Low'
    close: 'c',    // 기본: 'Close'
    volume: 'vol', // 기본: 'Volume'
  },
};
```

이동평균선 (sma / ema / volumeMA)

```JavaScript
const config = {
  sma: [
    { period: 20, enabled: true },
    { period: 60, enabled: false },
    { period: 120, enabled: false },
  ],
  ema: [
    { period: 20, enabled: false },
    { period: 50, enabled: false },
  ],
  volumeMA: [
    { period: 20, enabled: true },
  ],
};
```

VWAP

```JavaScript
const config = {
  vwap: [
    { anchor: 'session', enabled: true },
    { anchor: 'week', enabled: false },
    { anchor: 'month', enabled: false },
  ],
};
```

앵커 종류: `'session'` | `'week'` | `'month'` | `'YYYY-MM-DD'` (고정 날짜)

드로잉 기본 스타일

JavaScript

```
const config = {
  drawing: {
    strokeColor: '#e53935',
    strokeWidth: 1.5,
    fillColor: 'rgba(0,0,0,0)', // 투명
    dash: null,                 // 실선. 예: '6,3' → 파선
    opacity: 1.0,
    previewOpacity: 0.3,
  },
  drawingText: {
    color: '#e53935',
    fontSize: 13,
    bold: false,
  },
};
```

보조지표 파라미터

JavaScript

```
const config = {
  rsi: { period: 14, ob: 70, os: 30 },
  macd: { fast: 12, slow: 26, signal: 9 },
  stochastic: { kPeriod: 14, smooth: 3, dPeriod: 3, ob: 80, os: 20 },
  atr: { period: 14 },
  cci: { period: 14 },
  roc: { period: 12 },
  williamsR: { period: 14 },
  mfi: { period: 14 },
  bbWidth: { period: 20, multiplier: 2 },
  sar: { step: 0.02, max: 0.20 },
  ichimoku: { tenkan: 9, kijun: 26, senkou: 52, displacement: 26 },
};
```

가격 알림 (alert)

```JavaScript
const config = {
  alert: {
    enabled: true,
    alertTypes: ['horizontal', 'pricelabel', 'hzone'], // 알림 가능한 드로잉 타입
  },
};
```

템플릿 경로 (templates)

```JavaScript
const config = {
  templates: {
    drawingSetting: './template/drawingSetting.html',
    alertConfig: './template/alertConfig.html',
    priceInfo: './template/priceInfo.html',
    legend: './template/legend.html',
    subPanelLabel: './template/subPanelLabel.html',
    subPanelSetting: './template/subPanelSetting.html',
    dataMessage: './template/dataMessage.html',
  },
};
```

런타임에 설정 확인

```JavaScript
panel.getConfig();   // 현재 적용된 설정 반환
panel.getDefaults(); // 라이브러리 기본값 반환
```

04

데이터 형식 및 로드

기본 데이터 형식

내부적으로 사용하는 표준 필드명은 다음과 같습니다.

```JavaScript
{
  Date: '2025-06-01T09:00:00', // 일봉: 'YYYY-MM-DD', 분봉: 'YYYY-MM-DDTHH:mm:ss'
  Open: 68000,
  High: 69500,
  Low: 67800,
  Close: 69200,
  Volume: 123456,
}
```

필드명이 다를 경우 `fieldMap` 설정으로 매핑하세요. **3. 필드 매핑**

setData(data)

초기 또는 전체 데이터를 설정합니다. 기존 데이터는 교체됩니다.

```JavaScript
panel.setData(ohlcvArray);
```

리샘플링과 함께 사용

1분봉 원본 데이터를 저장하고 `rawInterval`로 표시 봉 단위를 지정합니다.

```JavaScript
// 1분봉을 저장하고 5분봉으로 표시
panel.setData(oneMinData, 1); // rawInterval = 1(분)
panel.setInterval(5);         // 5분봉으로 전환
panel.setInterval(15);        // 15분봉으로 전환
```

addData(data)

단일 틱 추가 (실시간)

```JavaScript
panel.addData({
  Date: '2025-06-01T09:02:38',
  Open: 69000,
  High: 69300,
  Low: 68900,
  Close: 69100,
  Volume: 3200,
});
```

- **같은 분**: 마지막 봉 OHLCV를 갱신합니다.
- **새 분**: 봉을 추가합니다.

배열 일괄 추가

```JavaScript
// 최신 데이터 추가 (뒤에 붙음)
panel.addData(newBarsArray);

// 과거 데이터 추가 (앞에 붙음)
panel.addData(oldBarsArray); // 자동으로 prepend 감지
```

과거 데이터 추가 시 현재 보기 위치가 자동으로 보정됩니다.

needData 이벤트로 지연 로드

스크롤이 데이터 끝에 닿으면 `needData` 이벤트가 발생합니다.

```JavaScript
document.getElementById('myChart').addEventListener('needData', (e) => {
  const { direction, date } = e.detail;
  // direction: 'backward' (과거) | 'forward' (미래)
  // date: 현재 표시 범위의 끝 날짜 문자열

  if (direction === 'backward') {
    fetchOlderData(date).then(data => panel.addData(data));
  }
});
```

setSymbol(info)

종목 정보를 설정합니다. 종목 변경 시 기존 데이터가 초기화되므로 `setData()` 전에 호출합니다.

```JavaScript
panel.setSymbol({
  caption: '삼성전자', // 화면 표시 이름
  symbol: '005930',   // 종목 코드
  decimal: 0,         // 가격 소수점 자릿수
  currency: 'KRW',    // 통화
});
```

setInterval(minutes)

`setData(data, rawInterval)`로 1분봉 원본을 먼저 로드한 경우에만 사용 가능합니다.

```JavaScript
panel.setInterval(1);  // 1분봉
panel.setInterval(5);  // 5분봉
panel.setInterval(60); // 60분봉(1시간봉)
```

setAxisMode(mode)

x축 표시 방식을 변경합니다.

```JavaScript
panel.setAxisMode('session');  // 세션 모드: 시간 갭 없음 (기본)
panel.setAxisMode('calendar'); // 캘린더 모드: 날짜 기준 x축
```

데이터 오류 처리

- **빈 배열**: `data-message-type="empty"` 오버레이를 표시합니다.
- **필드 누락**: `data-message-type="invalid"` 오버레이를 표시하고 누락 필드를 나열합니다.

오버레이 UI 커스터마이징은 **11. 템플릿 - dataMessage**를 참고하세요.

05

차트 마크 & 오버레이

마크(Mark)는 메인 차트에 렌더링되는 각종 시각 요소입니다.

기본 제공 마크

마크 이름	설명
`candle`	캔들스틱
`line`	라인 차트
`area`	영역 차트 (라인 + 음영)
`volume`	거래량 바
`bollinger`	볼린저 밴드
`highLow`	고가/저가 라인
`sma{period}`	단순이동평균 (예: `sma20`, `sma50`)
`ema{period}`	지수이동평균 (예: `ema20`, `ema50`)
`volumeMA{period}`	거래량 이동평균 (예: `volumeMA20`)
`vwap_{anchor}`	VWAP (예: `vwap_session`, `vwap_week`)
`sar`	Parabolic SAR
`ichimoku`	일목균형표

차트 타입 전환

JavaScript

```
panel.setCandleMode(); // 캔들스틱 (기본)
panel.setLineMode();   // 라인 차트
panel.setAreaMode();   // 영역 차트
```

마크 켜기/끄기

```JavaScript
panel.toggleMarks('volume');    // 거래량 바 토글
panel.toggleMarks('bollinger'); // 볼린저 밴드 토글
panel.toggleMarks('sma20');     // SMA 20 토글
panel.toggleMarks('highLow');   // 고가/저가 토글
```

SMA (단순이동평균)

```JavaScript
// 추가 (이미 있으면 활성화)
panel.addSmaMark(20);        // 기간 20, 활성화
panel.addSmaMark(60, false); // 기간 60, 비활성으로 추가

// 제거
panel.removeSmaMark(20);

// 토글
panel.toggleMarks('sma20');
```

EMA (지수이동평균)

```JavaScript
panel.addEmaMark(20);
panel.addEmaMark(50, false);
panel.removeEmaMark(20);
panel.toggleMarks('ema20');
```

거래량 이동평균

```JavaScript
panel.addVolumeMaMark(20);
panel.removeVolumeMaMark(20);
panel.toggleMarks('volumeMA20');
```

VWAP

```JavaScript
// 앵커 종류: 'session' | 'week' | 'month' | 'YYYY-MM-DD'
panel.addVwapMark('session');
panel.addVwapMark('week');
panel.addVwapMark('month');
panel.addVwapMark('2025-01-02'); // 고정 날짜 앵커

// 커스텀 스타일 적용
panel.addVwapMark('session', { strokeColor: '#2196f3', strokeWidth: 2 });

// 제거
panel.removeVwapMark('session');
panel.removeVwapMark('2025-01-02');
```

Parabolic SAR

```JavaScript
panel.addSarMark();           // 기본값 (step: 0.02, max: 0.20)
panel.addSarMark(0.01, 0.10); // 커스텀 파라미터
panel.toggleMarks('sar');     // 토글
```

일목균형표 (Ichimoku)

```JavaScript
panel.addIchimokuMark();
panel.addIchimokuMark({ tenkan: 9, kijun: 26, senkou: 52, displacement: 26 });
panel.toggleMarks('ichimoku');
```

비율 정규화 모드 (비교 차트)

여러 종목을 같은 시작점(%)으로 비교할 때 사용합니다.

JavaScript

```
panel.setNormalizeMode(true);  // % 변화율 y축으로 전환
panel.setNormalizeMode(false); // 절대 가격으로 복원

panel.isNormalizeMode();       // 현재 상태 반환 (boolean)
```

`PanelSync` 로 패널을 연동하면 자동으로 정규화 모드가 활성화됩니다.

표시 범위 조작

JavaScript

```
// 특정 날짜로 이동
panel.findDate('2025-05-15');

// 최근 데이터로 이동
panel.resetView();

// 표시 봉 개수 변경
panel.setDisplayCnt(100);

// 현재 보기 범위 조회
const range = panel.getVisibleRange();
// → { from: Date, to: Date, startIdx: number, displayCnt: number }

// 범위 지정 (외부 동기화용)
panel.setVisibleRange('2025-05-01', '2025-05-31');
```

06

보조지표 (Sub Panel)

서브패널은 메인 차트 하단에 독립된 패널로 표시되는 보조지표입니다.

지원 지표

지표 키	이름	파라미터
`rsi`	RSI (상대강도지수)	`period`
`macd`	MACD	`fast`, `slow`, `signal`
`stochastic`	Stochastic	`kPeriod`, `smooth`, `dPeriod`
`atr`	ATR (평균진폭범위)	`period`
`cci`	CCI (상품채널지수)	`period`
`roc`	ROC (변화율)	`period`
`williamsR`	Williams %R	`period`
`mfi`	MFI (자금흐름지수)	`period`
`obv`	OBV (누적거래량)	—
`bbWidth`	BB Width (볼린저밴드 폭)	`period`, `multiplier`

addSubPanel(options)

```JavaScript
panel.addSubPanel({ indicator: 'rsi', period: 14 });
panel.addSubPanel({ indicator: 'macd' }); // 기본 파라미터 사용
panel.addSubPanel({ indicator: 'macd', fast: 8, slow: 21, signal: 5 });
panel.addSubPanel({ indicator: 'stochastic', kPeriod: 14, smooth: 3, dPeriod: 3 });
panel.addSubPanel({ indicator: 'atr', period: 14 });
panel.addSubPanel({ indicator: 'cci', period: 20 });
panel.addSubPanel({ indicator: 'roc', period: 12 });
panel.addSubPanel({ indicator: 'williamsR', period: 14 });
panel.addSubPanel({ indicator: 'mfi', period: 14 });
panel.addSubPanel({ indicator: 'obv' });
panel.addSubPanel({ indicator: 'bbWidth', period: 20, multiplier: 2 });
```

서브패널은 메서드 체인이 가능합니다.

```JavaScript
new coForward_chart()
  .add('myChart')
  .setData(data)
  .addSubPanel({ indicator: 'rsi', period: 14 })
  .addSubPanel({ indicator: 'macd' });
```

removeSubPanels(indicator, period?)

```JavaScript
panel.removeSubPanels('rsi');     // RSI 서브패널 제거
panel.removeSubPanels('rsi', 14); // 기간 14인 RSI만 제거
panel.removeSubPanels('macd');
```

서브패널 높이

서브패널은 드래그 핸들로 높이를 조절할 수 있습니다.

- 최소 높이: `cfg.layout.subPanel.minHeight` (기본 60px)
- 최대 높이: `cfg.layout.subPanel.maxHeight` (기본 300px)
- 기본 높이: `cfg.layout.subPanel.height` (기본 100px)

초기 높이를 지정하려면:

```JavaScript
panel.addSubPanel({ indicator: 'rsi', period: 14, height: 150 });
```

또는 전체 기본값을 설정합니다.

```JavaScript
new coForward_chart({ layout: { subPanel: { height: 150 } } })
  .add('myChart')
  .setData(data);
```

서브패널 레이블 & 설정 UI

서브패널 상단 레이블과 설정 다이얼로그는 HTML 템플릿으로 제공됩니다.

- `template/subPanelLabel.html` : 이름, 파라미터 값, 닫기/설정 버튼
- `template/subPanelSetting.html` : 파라미터 변경 다이얼로그

자세한 내용은 **11. 템플릿**을 참고하세요.

07

드로잉 도구

사용자가 차트에 선, 도형, 텍스트를 직접 그릴 수 있는 도구입니다.

드로잉 타입 목록

```JavaScript
panel.getDrawingTypes(); // 사용 가능한 드로잉 타입 목록 반환
```

타입 키	카테고리	이름	클릭 횟수
`horizontal`	line	수평선	1
`vertical`	line	수직선	1
`trendline`	line	추세선	2
`extendedline`	line	연장선	2
`ray`	line	레이	2
`channel`	line	평행 채널	3
`pitchfork`	line	피치포크	3
`hzone`	shape	수평 구간	2
`rectangle`	shape	사각형	2
`measure`	shape	측정 도구	2
`fibonacci`	fibonacci	피보나치 되돌림	2
`fibonacciExt`	fibonacci	피보나치 확장	3
`pricelabel`	text	가격 레이블	1
`text`	text	텍스트	2
`freehand`	other	자유 곡선	드래그

드로잉 모드 설정

```JavaScript
// 드로잉 모드 활성화
panel.setDrawMode('trendline');
panel.setDrawMode('horizontal');
panel.setDrawMode('fibonacci');

// 드로잉 모드 해제
panel.setDrawMode('none');
```

모드 변경 시 `modeChange` 이벤트가 발생합니다. 8. 이벤트

드로잉 스타일 설정

개별 setter

```JavaScript
panel.setDrawingColor('#2196f3'); // 선 색상
panel.setDrawingWidth(2);         // 선 굵기
panel.setDrawingDash('6,3');      // 파선 패턴 (null: 실선)
panel.setDrawingFill('#f0f0f0');  // 채우기 색상
panel.setDrawingOpacity(0.8);     // 불투명도 (0~1)
```

텍스트 스타일

```JavaScript
panel.setTextColor('#ffffff');
panel.setTextSize(14);
panel.setTextBold(true);
```

그리기 기본설정 다이얼로그

```JavaScript
panel.showDrawingSettings();
```

UI 다이얼로그로 선 색상, 굵기, 파선, 채우기 색상, 텍스트 스타일을 설정합니다.
`template/drawingSetting.html` 템플릿이 필요합니다.

드로잉 조작

키보드 단축키

키	동작
`Delete` / `Backspace`	선택된 드로잉 삭제
`Escape`	드로잉 모드 해제 / 선택 취소

드로잉 초기화

```JavaScript
panel.clearDrawings(); // 모든 드로잉 삭제 (마크/지표는 유지)
```

가격 알림 연동

`horizontal`, `pricelabel`, `hzone` 드로잉 타입은 가격 알림을 설정할 수 있습니다. 해당 드로잉을 클릭하면 알림 설정 다이얼로그가 표시됩니다.

알림 방향 옵션

드로잉 타입	direction 값	설명
horizontal, pricelabel	`'up'`	상향 돌파
	`'down'`	하향 돌파
	`'both'`	양방향

드로잉 타입	direction 값	설명
hzone	`'enter_up'`	하단에서 진입
	`'enter_down'`	상단에서 진입
	`'enter'`	어느 방향이든 진입
	`'exit_up'`	상단으로 이탈
	`'exit_down'`	하단으로 이탈
	`'exit'`	어느 방향이든 이탈
	`'both'`	모든 교차

알림 모드

mode	설명
`'once'`	1회 발동 후 비활성화
`'repeat'`	`cooldown`(초) 간격으로 반복 발동

알림 API (코드에서 직접 설정)

```JavaScript
// 알림 설정
panel.setAlert(drawingIdx, {
  enabled: true,
  direction: 'up',
  mode: 'repeat',
  cooldown: 60, // 반복 간격(초)
  label: '상향 돌파',
});

// 알림 조회
const alertInfo = panel.getAlert(drawingIdx);

// 알림 삭제
panel.clearAlert(drawingIdx);
```

priceAlert 이벤트

```JavaScript
document.getElementById('myChart').addEventListener('priceAlert', (e) => {
  const { symbol, price, target, direction, label, triggeredAt } = e.detail;
  console.log(`[알림] ${symbol}: ${price} ${direction} ${target}`);
});
```

08

이벤트 시스템

차트는 두 가지 방법으로 이벤트를 구독할 수 있습니다.

이벤트 등록 방법

방법 1: DOM 이벤트 (컨테이너 기준)

```JavaScript
document.getElementById('myChart').addEventListener('needData', (e) => {
  console.log(e.detail);
});
```

방법 2: 인스턴스 메서드 .on() / .off()

```JavaScript
// 등록
panel.on('crosshairMove', (e) => {
  console.log(e.detail.date, e.detail.candle);
});

// 해제
panel.off('crosshairMove', handler);
```

needData

스크롤이 데이터 끝에 닿았을 때 발생합니다. 추가 데이터를 서버에서 불러오는 데 사용합니다.

```JavaScript
panel.on('needData', (e) => {
  const { direction, date } = e.detail;
  // direction: 'backward' (과거) | 'forward' (미래)
  // date: 현재 표시 범위 끝 날짜 문자열
});
```

rangeChange

스크롤/줌으로 표시 범위가 변경될 때 발생합니다.

```JavaScript
panel.on('rangeChange', (e) => {
  const { from, to, startIdx, displayCnt } = e.detail;
  // from, to: Date 객체
});
```

crosshairMove

마우스 이동 시 십자선 위치와 해당 봉 정보를 전달합니다.

```JavaScript
panel.on('crosshairMove', (e) => {
  const { date, candle } = e.detail;
  // date: 날짜 문자열
  // candle: { Date, Open, High, Low, Close, Volume }
});
```

crosshairClick

차트 클릭 시 해당 봉 정보를 전달합니다.

```JavaScript
panel.on('crosshairClick', (e) => {
  const { date, candle } = e.detail;
});
```

priceAlert

가격 알림 조건이 충족되면 발생합니다.

```JavaScript
panel.on('priceAlert', (e) => {
  const { symbol, price, target, direction, label, drawingIdx, triggeredAt } = e.detail;
  // direction: 'up' | 'down'
  alert(`${symbol} 가격 알림: ${price} (${direction})`);
});
```

modeChange

드로잉 모드가 변경될 때 발생합니다.

```JavaScript
panel.on('modeChange', (e) => {
  const { mode } = e.detail;
  // mode: 드로잉 타입 문자열 | 'none'
  console.log('현재 모드:', mode);
});
```

십자선 제어 (외부 동기화)

```JavaScript
// 특정 날짜에 십자선 표시
panel.showCrosshairAtDate('2025-06-01T09:30:00');

// 십자선 숨김
panel.hideCrosshair();
```

> `PanelSync` 를 사용하면 연동된 패널 간에 십자선이 자동으로 동기화됩니다. **10. PanelSync**

09

상태 저장 & 복원

드로잉, 지표, 서브패널, 드로잉 스타일을 저장하고 복원합니다.

localStorage 자동 저장

`autoSave: true` (기본값)이면 드로잉 추가/삭제/수정 시 자동으로 localStorage에 저장됩니다.

저장 키 형식: `{autoSaveKeyPrefix}_{symbol}_{interval}`
예) `coForward_chart2_005930_1`

수동 저장/복원

```JavaScript
panel.saveToLocal();   // 현재 상태를 localStorage에 저장
panel.loadFromLocal(); // localStorage에서 상태 복원
panel.clearLocal();    // localStorage에서 해당 상태 항목 삭제
```

페이지 로드 시 자동 복원 예시

```JavaScript
const panel = new coForward_chart()
  .add('myChart')
  .setSymbol({ symbol: '005930', decimal: 0, currency: 'KRW' })
  .setData(data);

panel.loadFromLocal(); // setData() 이후에 호출
```

JSON 직렬화 저장/복원

서버 저장 또는 공유가 필요한 경우 JSON 문자열로 변환합니다.

JavaScript

```javascript
// 저장
const json = panel.saveState();
// → JSON 문자열 (drawings, indicators, subPanels, drawingStyle 포함)

// 복원
panel.loadState(json);

// 드로잉만 복원
panel.loadState(json, { drawingsOnly: true });
```

saveState() 반환 형식 예시

JSON

```json
{
  "version": 1,
  "symbol": "005930",
  "timeframe": 1,
  "savedAt": "2025-06-01T09:00:00.000Z",
  "indicators": {
    "marks": [
      { "name": "candle", "enabled": true },
      { "name": "bollinger", "enabled": false }
    ],
    "sma": [{ "period": 20, "enabled": true }],
    "ema": [],
    "volumeMA": [],
    "vwap": []
  },
  "subPanels": [
    { "indicator": "rsi", "period": 14 }
  ],
  "drawings": [ ... ],
  "drawingStyle": {
    "drawing": { "strokeColor": "#e53935", "strokeWidth": 1.5, ... },
    "drawingText": { "color": "#e53935", "fontSize": 13, "bold": false }
  }
}
```

상태 초기화

```JavaScript
// 드로잉만 초기화
panel.clearDrawings();

// 드로잉 + 마크 모두 초기화 (config 기본값으로 복원)
panel.resetState();

// 개별 초기화 옵션
panel.resetState({ drawings: true, markers: false }); // 드로잉만
panel.resetState({ drawings: false, markers: true }); // 마크만
```

자동 저장 제어

```JavaScript
panel.autoSave(false); // 자동 저장 비활성화
panel.autoSave(true);  // 자동 저장 활성화
```

10

패널 동기화 (PanelSync)

여러 `coForward_chart` 인스턴스의 스크롤 범위와 십자선을 동기화합니다. 패널 연동 시 자동으로 **% 정규화 모드**가 활성화되어 비교 차트로 사용할 수 있습니다.

가져오기

JavaScript

```
import { coForward_chart, PanelSync } from '/dist/coForward_chart2.min.js';
```

기본 사용법

JavaScript

```
const panel1 = new coForward_chart().add('chart1').setData(data1);
const panel2 = new coForward_chart().add('chart2').setData(data2);
const panel3 = new coForward_chart().add('chart3').setData(data3);

const sync = new PanelSync();

// 패널 추가 (단일 또는 배열)
sync.add(panel1, panel2);
sync.add(panel3); // 나중에 추가 가능

// 특정 패널만 제거 (패널 상태 복원됨)
sync.remove(panel2);

// 전체 해제
sync.removeAll();

// 전체 해제 + 인스턴스 정리
sync.destroy();
```

동기화 내용

항목	설명
스크롤/줌 범위	`rangeChange` 이벤트 기반으로 동일 날짜 범위 유지

항목	설명
십자선 위치	`crosshairMove` 이벤트 기반으로 동일 위치 표시
% 정규화 모드	패널 추가 시 자동 활성화, 제거 시 자동 복원

주의사항

- 서로 다른 timeframe의 패널을 연동할 경우, 날짜 범위 매칭이 정확하지 않을 수 있습니다.
- `setData()` 이전에 `sync.add()`를 호출하면 초기 범위 동기화가 되지 않습니다. `setData()` 이후에 추가하세요.

11

HTML 템플릿 커스터마이징

차트의 오버레이 UI(범례, 가격 정보, 알림 설정 등)는 외부 HTML 파일로 분리되어 있습니다. 템플릿 파일이 없으면 해당 기능이 비활성화됩니다(오류 없음).

템플릿 경로 설정

기본 경로는 `./template/*.html` 입니다. `config.templates` 로 변경할 수 있습니다.

```JavaScript
new coForward_chart({
  templates: {
    drawingSetting: '/assets/templates/drawingSetting.html',
    alertConfig: '/assets/templates/alertConfig.html',
    priceInfo: '/assets/templates/priceInfo.html',
    legend: '/assets/templates/legend.html',
    subPanelLabel: '/assets/templates/subPanelLabel.html',
    subPanelSetting: '/assets/templates/subPanelSetting.html',
    dataMessage: '/assets/templates/dataMessage.html',
  },
})
```

템플릿 바인딩 속성 (data-coforward-*)

라이브러리는 `data-coforward-*` 속성으로 DOM 요소를 찾아 값을 주입합니다. 속성명을 유지하면 구조(태그, 클래스)는 자유롭게 변경할 수 있습니다.

priceInfo.html: 십자선 가격 정보 패널

속성	설명
`data-coforward-price-info-caption`	종목 표시 이름
`data-coforward-price-info-symbol`	종목 코드
`data-coforward-price-info-ohlcv="open"`	시가
`data-coforward-price-info-ohlcv="high"`	고가

속성	설명
`data-coforward-price-info-ohlcv="low"`	저가
`data-coforward-price-info-ohlcv="close"`	종가
`data-coforward-price-info-ohlcv="volume"`	거래량

HTML

```
<article class="coForward-chart2-price-info">
  <h1>
    <span data-coforward-price-info-caption></span>
    <span data-coforward-price-info-symbol></span>
  </h1>
  <ul>
    <li>시가 <span data-coforward-price-info-ohlcv="open"></span></li>
    <li>고가 <span data-coforward-price-info-ohlcv="high"></span></li>
    <li>저가 <span data-coforward-price-info-ohlcv="low"></span></li>
    <li>종가 <span data-coforward-price-info-ohlcv="close"></span></li>
    <li>거래량 <span data-coforward-price-info-ohlcv="volume"></span></li>
  </ul>
</article>
```

legend.html: 마크 범례

속성	설명
`data-coforward-legend-title`	범례 제목
`data-coforward-legend-list`	항목 목록 컨테이너
`data-coforward-legend-item`	반복 항목 템플릿 (JS에서 `cloneNode` 사용)
`data-coforward-legend-label`	마크 이름
`data-coforward-legend-swatch`	색상 견본 SVG
`data-coforward-legend-line`	색상 견본 선 요소
`data-coforward-legend-circle`	색상 견본 점 요소
`data-coforward-legend-remove`	제거 버튼

subPanelLabel.html: 서브패널 레이블

속성	설명
`data-coforward-subpanel-name`	지표 이름
`data-coforward-subpanel-period`	기간 파라미터
`data-subpanel-values`	현재값 목록 컨테이너
`data-subpanel-value-item`	반복 값 항목 템플릿
`data-subpanel-value-label`	값 레이블
`data-subpanel-value-num`	값 숫자
`data-subpanel-close`	닫기 버튼
`data-subpanel-settings`	설정 버튼

subPanelSetting.html: 서브패널 설정 다이얼로그

파라미터 변경 다이얼로그입니다. `name` 속성으로 입력값을 수집합니다.

HTML

```
<div class="subPanel-setting-dialog">
  <form>
    <div data-subpanel-setting-item="period">
      <label>기간</label>
      <input type="number" name="period" min="1">
    </div>
    <footer>
      <button type="button" class="btn-cancel">취소</button>
      <button type="button" class="btn-submit">확인</button>
    </footer>
  </form>
</div>
```

drawingSetting.html: 그리기 기본설정 다이얼로그

`name` 속성으로 값을 읽고 씁니다. SVG 미리보기에는 `data-sample-*` 속성을 사용합니다.

name 값	설명
`strokeColor`	선 색상 (`type="color"`)
`strokeWidth`	선 굵기
`dasharray`	파선 패턴 (빈 값: 실선)
`fillColor`	채우기 색상 (`type="color"`)
`textColor`	텍스트 색상 (`type="color"`)
`fontSize`	텍스트 크기
`bold`	굵게 (`'true'` / `'false'`)

SVG 속성	설명
`data-sample-rect`	미리보기 사각형
`data-sample-text`	미리보기 텍스트

버튼 클래스	역할
`.btn-submit`	확인 (설정 저장)
`.btn-cancel`	취소

alertConfig.html: 가격 알림 설정 다이얼로그

속성	설명
`data-coforward-chart-drawing`	드로잉 타입 이름 표시
`data-coforward-alert-item="label"`	알림 이름 입력
`data-coforward-alert-item="price"`	가격 (readonly)
`data-coforward-alert-item="direction"`	방향 선택
`data-coforward-alert-item="mode"`	알림 방식 선택 (`once` / `repeat`)
`data-coforward-alert-item="cooldown"`	반복 간격(초) 입력

버튼 클래스	역할
`.btn-submit`	확인 (알림 저장)
`.btn-cancel`	취소
`.btn-delete`	알림 삭제

dataMessage.html: 데이터 상태 오버레이

데이터가 없거나 유효하지 않을 때 차트 위에 표시됩니다.

속성	표시 조건
`data-message-type="empty"`	데이터 배열이 비어있을 때
`data-message-type="loading"`	로딩 중 (외부에서 직접 호출)
`data-message-type="invalid"`	필수 필드가 누락된 데이터일 때

`invalid` 타입에서 누락 필드 목록은 `data-message-detail` 요소에 주입됩니다.

HTML

```
<div class="cf-empty-overlay">
  <div data-message-type="empty">데이터가 없습니다.</div>
  <div data-message-type="invalid">
    잘못된 데이터 형식입니다.
    <ul data-message-detail></ul>
  </div>
</div>
```

12

전체 API 레퍼런스

생성자

```JavaScript
new coForward_chart(config?)
```

`config` 는 선택 사항입니다. **3. Config**

초기화

메서드	반환	설명
`.add(containerId)`	`this`	지정 ID의 컨테이너에 차트 마운트
`.setSymbol({ caption, symbol, decimal, currency })`	`this`	종목 정보 설정 (데이터 초기화됨)

데이터

메서드	반환	설명
`.setData(data, rawInterval?)`	`this`	전체 데이터 설정
`.addData(data)`	`this`	단일 틱 또는 배열 추가 (실시간/지연로드)
`.setRawData(data, rawInterval)`	`this`	1분봉 원본 저장 + 리샘플링 표시
`.setInterval(minutes)`	`boolean`	봉 단위 전환 (`_rawData` 필요)
`.setAxisMode('session'|'calendar')`	`this`	x축 모드 변경

렌더링 & 뷰

메서드	반환	설명
`.render(startIdx?, displayCnt?)`	`this`	수동 렌더링

메서드	반환	설명
`.setDisplayCnt(count)`	`void`	표시 봉 개수 변경
`.resetView()`	`this`	최신 데이터로 이동
`.findDate(date)`	`this`	특정 날짜로 이동 (툴팁도 표시)
`.getVisibleRange()`	`{from, to, startIdx, displayCnt}`	현재 표시 범위 반환
`.setVisibleRange(from, to)`	`this`	날짜 범위로 표시 범위 설정

차트 타입 & 마크

메서드	반환	설명
`.setCandleMode()`	`this`	캔들스틱 모드
`.setLineMode()`	`this`	라인 차트 모드
`.setAreaMode()`	`this`	영역 차트 모드
`.toggleMarks(name)`	`this`	마크 표시 토글
`.addSmaMark(period, enabled?)`	`this`	SMA 추가/활성화
`.removeSmaMark(period)`	`this`	SMA 제거
`.addEmaMark(period, enabled?)`	`this`	EMA 추가/활성화
`.removeEmaMark(period)`	`this`	EMA 제거
`.addVolumeMaMark(period, enabled?)`	`this`	거래량 이동평균 추가
`.removeVolumeMaMark(period)`	`this`	거래량 이동평균 제거
`.addSarMark(step?, max?)`	`this`	Parabolic SAR 추가
`.addIchimokuMark(cfg?)`	`this`	일목균형표 추가
`.addVwapMark(anchor, style?)`	`this`	VWAP 추가
`.removeVwapMark(anchor)`	`this`	VWAP 제거
`.setNormalizeMode(enabled)`	`this`	% 정규화 모드 설정
`.isNormalizeMode()`	`boolean`	현재 정규화 모드 상태

서브패널

메서드	반환	설명
`.addSubPanel(options)`	`this`	보조지표 서브패널 추가
`.removeSubPanels(indicator, period?)`	`this`	서브패널 제거

options 키

```JavaScript
{
  indicator: 'rsi', // 지표 키
  period: 14,       // 기간
  fast: 12,         // MACD fast
  slow: 26,         // MACD slow
  signal: 9,        // MACD signal
  kPeriod: 14,      // Stochastic %K
  smooth: 3,        // Stochastic smooth
  dPeriod: 3,       // Stochastic %D
  multiplier: 2,    // BBWidth multiplier
  height: 100,      // 초기 높이(px)
}
```

드로잉

메서드	반환	설명
`.setDrawMode(type)`	`this`	드로잉 모드 설정 (`'none'` 으로 해제)
`.showDrawingSettings()`	`void`	그리기 기본설정 다이얼로그 표시
`.setDrawingColor(color)`	`this`	선 색상
`.setDrawingWidth(width)`	`this`	선 굵기
`.setDrawingDash(dash)`	`this`	파선 패턴 (null: 실선)
`.setDrawingFill(color)`	`this`	채우기 색상
`.setDrawingOpacity(opacity)`	`this`	불투명도
`.setDrawingStyle(style)`	`this`	드로잉 스타일 객체 일괄 설정

메서드	반환	설명
`.setTextColor(color)`	`this`	텍스트 색상
`.setTextSize(size)`	`this`	텍스트 크기
`.setTextBold(bold)`	`this`	텍스트 굵기
`.setTextStyle(style)`	`this`	텍스트 스타일 객체 일괄 설정
`.clearDrawings()`	`this`	모든 드로잉 삭제
`.getDrawingTypes()`	`object`	드로잉 타입 목록 반환

가격 알림

메서드	반환	설명
`.setAlert(drawingIdx, options)`	`this`	드로잉에 알림 설정
`.getAlert(drawingIdx)`	`object`	알림 정보 조회
`.clearAlert(drawingIdx)`	`this`	알림 삭제
`.setAlertConfig(options)`	`this`	알림 전체 설정 변경

상태 저장/복원

메서드	반환	설명
`.saveState()`	`string`	상태 JSON 문자열 반환
`.loadState(json, options?)`	`boolean`	JSON에서 상태 복원
`.saveToLocal()`	`this`	localStorage에 저장
`.loadFromLocal()`	`this`	localStorage에서 복원
`.clearLocal()`	`this`	localStorage 항목 삭제
`.autoSave(enabled)`	`this`	자동 저장 활성화/비활성화
`.resetState(options?)`	`this`	상태 초기화

resetState 옵션

```JavaScript
{ drawings: true, markers: true } // 기본: 모두 초기화
```

이벤트

메서드	반환	설명
`.on(eventName, handler)`	`this`	이벤트 등록
`.off(eventName, handler)`	`this`	이벤트 해제

이벤트	detail	설명
`needData`	`{ direction, date }`	추가 데이터 요청
`rangeChange`	`{ from, to, startIdx, displayCnt }`	표시 범위 변경
`crosshairMove`	`{ date, candle }`	십자선 이동
`crosshairClick`	`{ date, candle }`	차트 클릭
`priceAlert`	`{ symbol, price, target, direction, label, drawingIdx, triggeredAt }`	가격 알림 발동
`modeChange`	`{ mode }`	드로잉 모드 변경

십자선

메서드	반환	설명
`.showCrosshairAtDate(date)`	`void`	날짜에 십자선 표시
`.hideCrosshair()`	`this`	십자선 숨김

기타

메서드	반환	설명
`.saveToImage(options?)`	`Promise`	차트를 이미지로 저장
`.getConfig()`	`object`	현재 설정 반환
`.getDefaults()`	`object`	라이브러리 기본값 반환
`.destroy()`	`void`	인스턴스 정리 (이벤트 해제, DOM 제거)

PanelSync 클래스

```JavaScript
import { PanelSync } from '/dist/coForward_chart2.min.js';
```

메서드	반환	설명
`new PanelSync()`	—	인스턴스 생성
`.add(...panels)`	`this`	패널 연동 추가
`.remove(...panels)`	`this`	패널 연동 해제
`.removeAll()`	`this`	전체 연동 해제
`.destroy()`	`void`	전체 해제 + 정리