<!DOCTYPE html>

<html lang="zh-CN">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Heark 虚拟货币交易所</title>

<script src="https://cdn.jsdelivr.net/npm/lightweight-charts@3.8.0/dist/lightweight-charts.standalone.production.js"></script>

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css">

<style>

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

font-family: 'Arial', sans-serif;

}

body {

background-color: #000000;

color: #eaecef;

line-height: 1.6;

overflow-x: hidden;

}

.container {

max-width: 1800px;

margin: 0 auto;

padding: 20px;

}

header {

display: flex;

justify-content: space-between;

align-items: center;

padding: 10px 0;

border-bottom: 1px solid #1c2333;

margin-bottom: 20px;

position: relative;

}

.logo {

font-size: 32px;

font-weight: bold;

color: #2a9df4;

text-transform: uppercase;

letter-spacing: 2px;

text-shadow: 0 0 10px rgba(42, 157, 244, 0.5);

}

.logo::after {

content: '™';

font-size: 0.5em;

vertical-align: super;

}

.control-button {

background: #1c2333;

color: #2a9df4;

border: 1px solid #2a3040;

padding: 8px 15px;

border-radius: 5px;

cursor: pointer;

display: flex;

align-items: center;

gap: 5px;

}

.main-content {

display: grid;

grid-template-columns: 250px 1fr 350px;

grid-template-rows: 550px 1fr;

gap: 20px;

margin-bottom: 20px;

}

.left-panel {

grid-row: 1 / 3;

grid-column: 1;

background: #131722;

border-radius: 10px;

padding: 15px;

min-height: 800px;

overflow-y: auto;

}

.coin-select-left {

margin-bottom: 20px;

}

.coin-option {

display: flex;

justify-content: space-between;

padding: 8px 15px;

background: #131722;

border-radius: 5px;

cursor: pointer;

border: 1px solid #2a3040;

margin-bottom: 10px;

}

.coin-option.active {

background: #2a3040;

border-color: #2a9df4;

}

.coin-price-small {

font-size: 14px;

font-weight: bold;

}

.coin-change-small {

font-size: 12px;

}

.price-up-small { color: #00b15d; }

.price-down-small { color: #ff5b5a; }

.price-up { color: #00b15d; }

.price-down { color: #ff5b5a; }

.coin-header .price-up, .coin-header .price-down-small { color: #00b15d; }

.coin-header .price-down, .coin-header .price-down-small { color: #ff5b5a; }

.chart-container {

grid-row: 1;

grid-column: 2;

background: #131722;

border-radius: 10px;

padding: 15px;

height: 550px;

position: relative;

z-index: 1;

}

.chart-controls {

position: absolute;

top: 10px;

right: 10px;

z-index: 10;

}

.fit-view-btn {

background: #2a9df4;

color: white;

border: none;

padding: 5px 10px;

border-radius: 3px;

cursor: pointer;

font-size: 12px;

}

.coin-header {

display: flex;

align-items: center;

gap: 15px;

margin-bottom: 10px;

}

.timeframe-selector {

display: flex;

gap: 5px;

margin-bottom: 15px;

}

.timeframe-btn {

padding: 5px 10px;

background: #1c2333;

border: 1px solid #2a3040;

border-radius: 3px;

cursor: pointer;

font-size: 12px;

color: #9aa0a6;

}

.timeframe-btn.active {

background: #2a3040;

color: #2a9df4;

border-color: #2a9df4;

}

.order-book {

grid-row: 1 / 3;

grid-column: 3;

background: #131722;

border-radius: 10px;

padding: 15px;

min-height: 800px;

overflow-y: auto;

position: relative;

}

.trade-panel {

grid-row: 2;

grid-column: 2;

background: #131722;

border-radius: 10px;

padding: 15px;

margin-bottom: 20px;

height: auto;

}

.balance-info {

background: #131722;

border-radius: 10px;

padding: 15px;

margin-top: 10px;

}

.panel-title {

font-size: 16px;

font-weight: bold;

}

.trade-mode {

display: flex;

gap: 10px;

margin-bottom: 10px;

}

.mode-option {

padding: 5px 10px;

background: #1c2333;

border-radius: 3px;

cursor: pointer;

font-size: 12px;

}

.mode-option.active {

background: #2a9df4;

color: #fff;

}

.position-buttons {

display: none;

gap: 5px;

margin-bottom: 10px;

}

.position-btn {

padding: 5px 10px;

background: #1c2333;

border-radius: 3px;

cursor: pointer;

font-size: 12px;

}

.leverage-selector {

display: none;

gap: 5px;

margin-bottom: 10px;

}

.leverage-option {

padding: 5px 10px;

background: #1c2333;

border-radius: 3px;

cursor: pointer;

font-size: 12px;

}

.leverage-option.active {

background: #2a9df4;

color: #fff;

}

.contract-mode-buttons {

display: none;

gap: 10px;

margin-bottom: 10px;

}

.contract-mode-btn {

flex: 1;

padding: 8px;

border-radius: 5px;

cursor: pointer;

font-weight: bold;

border: none;

color: white;

transition: background 0.3s;

}

.contract-mode-btn.active {

background: #666 !important;

}

.open-position-btn {

background: #00b15d;

}

.open-position-btn:hover:not(.active) {

background: #00934a;

}

.close-position-btn {

background: #ff5b5a;

}

.close-position-btn:hover:not(.active) {

background: #e04443;

}

.input-group {

margin-bottom: 15px;

position: relative;

}

.input-group.stop-loss,

.input-group.take-profit {

display: block;

}

.input-group.hidden {

display: none;

}

.input-group label {

display: block;

margin-bottom: 5px;

color: #9aa0a6;

font-size: 14px;

}

.input-group input {

width: 100%;

padding: 10px;

background: #1c2333;

border: 1px solid #2a3040;

border-radius: 5px;

color: #eaecef;

}

#copy-price {

position: absolute;

right: 5px;

top: 35px;

padding: 5px 10px;

background: #2a9df4;

color: white;

border: none;

border-radius: 3px;

cursor: pointer;

font-size: 12px;

}

.action-buttons {

display: grid;

grid-template-columns: 1fr 1fr;

gap: 10px;

margin-top: 20px;

}

.buy-btn, .sell-btn {

padding: 12px;

border-radius: 5px;

font-weight: bold;

cursor: pointer;

border: none;

transition: background 0.3s;

}

.buy-btn {

background: #00b15d;

color: white;

}

.buy-btn:hover {

background: #00934a;

}

.sell-btn {

background: #ff5b5a;

color: white;

}

.sell-btn:hover {

background: #e04443;

}

.order-book-title {

font-size: 16px;

font-weight: bold;

margin-bottom: 15px;

}

.order-table, .history-table {

width: 100%;

border-collapse: collapse;

}

.order-table th, .order-table td, .history-table th, .history-table td {

padding: 8px 5px;

text-align: right;

font-size: 13px;

}

.order-table th:first-child, .order-table td:first-child, .history-table th:first-child, .history-table td:first-child {

text-align: left;

}

.bid-price { color: #00b15d; }

.ask-price { color: #ff5b5a; }

.order-book-tabs {

display: flex;

gap: 10px;

}

.order-book-tab {

padding: 5px 10px;

background: #1c2333;

border-radius: 3px;

cursor: pointer;

font-size: 12px;

}

.order-book-tab.active {

background: #2a3040;

color: #2a9df4;

}

.trade-history {

margin-top: 20px;

}

.history-buy { color: #00b15d; }

.history-sell { color: #ff5b5a; }

.positions-section, .history-section {

background: #131722;

border-radius: 10px;

padding: 15px;

margin-bottom: 10px;

}

.positions-table, .history-completed-table {

width: 100%;

border-collapse: collapse;

}

.positions-table th, .positions-table td, .history-completed-table th, .history-completed-table td {

padding: 8px 5px;

text-align: left;

font-size: 12px;

}

.warning {

text-align: center;

color: #ff5b5a;

font-size: 12px;

margin-top: 10px;

}

footer {

text-align: center;

padding: 20px;

color: #9aa0a6;

border-top: 1px solid #1c2333;

margin-top: 30px;

font-size: 14px;

}

.modal {

display: none;

position: absolute;

top: 0;

left: 0;

width: 350px;

background: rgba(0, 0, 0, 0.8);

z-index: 1000;

}

.modal-content {

background: #131722;

border-radius: 10px;

padding: 20px;

width: 100%;

height: 100%;

overflow-y: auto;

}

.modal-header {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 20px;

border-bottom: 1px solid #2a3040;

padding-bottom: 10px;

}

.close-modal {

background: none;

border: none;

color: #9aa0a6;

font-size: 20px;

cursor: pointer;

}

.modal-controls {

display: flex;

flex-direction: column;

gap: 15px;

}

.modal-control {

display: flex;

justify-content: space-between;

align-items: center;

}

.modal-control input {

width: 100px;

padding: 8px;

background: #1c2333;

border: 1px solid #2a3040;

border-radius: 5px;

color: #eaecef;

}

.modal-buttons {

display: flex;

justify-content: flex-end;

gap: 10px;

margin-top: 20px;

}

.modal-btn {

padding: 8px 15px;

border-radius: 5px;

cursor: pointer;

border: none;

}

.modal-btn-primary {

background: #2a9df4;

color: white;

}

.modal-btn-secondary {

background: #1c2333;

color: #9aa0a6;

border: 1px solid #2a3040;

}

.pause-btn {

background: #ff5b5a;

color: white;

padding: 8px 15px;

border-radius: 5px;

cursor: pointer;

border: none;

}

.pause-btn.active {

background: #00b15d;

}

.success-message {

color: #00b15d;

font-size: 12px;

margin-top: 10px;

display: none;

}

@media (max-width: 1200px) {

.main-content {

grid-template-columns: 1fr;

grid-template-rows: auto;

}

.left-panel, .chart-container, .order-book, .trade-panel {

grid-row: auto;

grid-column: auto;

}

}

@media (max-width: 768px) {

header {

flex-direction: column;

gap: 15px;

}

}

</style>

</head>

<body>

<div class="container">

<header>

<div class="logo">Heark</div>

<button class="control-button" id="control-panel-btn">

<i class="fas fa-cog"></i> 控制面板

</button>

</header>

<div class="main-content">

<div class="left-panel">

<div class="coin-select-left">

<h3>币种</h3>

<div class="coin-option active" data-coin="HRK">

<span>HRK/USDT</span>

<div>

<div class="coin-price-small" id="hrk-price-small">$5000.00</div>

<div class="coin-change-small price-up-small" id="hrk-change-small">+0.00%</div>

</div>

</div>

<div class="coin-option" data-coin="NJC">

<span>NJC/USDT</span>

<div>

<div class="coin-price-small" id="njc-price-small">$1000.00</div>

<div class="coin-change-small price-up-small" id="njc-change-small">+0.00%</div>

</div>

</div>

</div>

</div>

<div class="chart-container">

<div class="chart-controls">

<button class="fit-view-btn" id="fit-view-btn">定位</button>

</div>

<div class="coin-header">

<h1 id="current-coin">HRK/USDT</h1>

<div id="current-price" class="coin-price price-up">$5000.00</div>

<div id="price-change" class="price-change price-up">+0.00%</div>

</div>

<div class="timeframe-selector">

<div class="timeframe-btn" data-tf="1">1分</div>

<div class="timeframe-btn" data-tf="5">5分</div>

<div class="timeframe-btn active" data-tf="15">15分</div>

<div class="timeframe-btn" data-tf="30">30分</div>

<div class="timeframe-btn" data-tf="60">1小时</div>

<div class="timeframe-btn" data-tf="240">4小时</div>

<div class="timeframe-btn" data-tf="D">日线</div>

</div>

<div id="chart"></div>

</div>

<div class="order-book">

<div class="order-book-header">

<div class="order-book-title">订单簿</div>

<div class="order-book-tabs">

<div class="order-book-tab active" id="order-tab">订单簿</div>

<div class="order-book-tab" id="trade-tab">最新成交</div>

</div>

</div>

<table class="order-table" id="order-table">

<thead>

<tr>

<th>价格 (USDT)</th>

<th>数量</th>

<th>总计</th>

</tr>

</thead>

<tbody id="order-book-body">

</tbody>

</table>

<div class="trade-history" id="trade-history" style="display: none;">

<div class="order-book-title">最新成交</div>

<table class="history-table">

<thead>

<tr>

<th>时间</th>

<th>价格</th>

<th>数量</th>

<th>方向</th>

</tr>

</thead>

<tbody id="trade-history-body">

</tbody>

</table>

</div>

</div>

<div class="trade-panel">

<div class="panel-header">

<div class="panel-title">交易 <span id="trade-pair">HRK/USDT</span></div>

<div class="trade-mode">

<div class="mode-option active" data-mode="spot">现货</div>

<div class="mode-option" data-mode="contract">合约</div>

</div>

</div>

<div class="contract-mode-buttons">

<button class="contract-mode-btn open-position-btn active" id="open-position-btn">开仓</button>

<button class="contract-mode-btn close-position-btn" id="close-position-btn">平仓</button>

</div>

<div class="leverage-selector" style="display: none;">

<div class="leverage-option active" data-value="1">1x</div>

<div class="leverage-option" data-value="3">3x</div>

<div class="leverage-option" data-value="5">5x</div>

<div class="leverage-option" data-value="10">10x</div>

<div class="leverage-option" data-value="20">20x</div>

</div>

<div class="input-group">

<label for="price">价格 (USDT)</label>

<input type="text" id="price" value="5000.00">

<button id="copy-price">复制当前市价</button>

</div>

<div class="input-group">

<label for="amount">数量</label>

<input type="number" id="amount" value="0.01" min="0.01" step="0.01">

</div>

<div class="input-group stop-loss">

<label for="stop-loss">止损价格</label>

<input type="number" id="stop-loss" placeholder="设置止损价格" step="0.01">

</div>

<div class="input-group take-profit">

<label for="take-profit">止盈价格</label>

<input type="number" id="take-profit" placeholder="设置止盈价格" step="0.01">

</div>

<div class="action-buttons">

<button class="buy-btn" id="buy-btn">买入</button>

<button class="sell-btn" id="sell-btn">卖出</button>

</div>

<div class="balance-info" id="long-balance">

<div class="data-title">多倉價值</div>

<div class="balance-value" id="long-value">0.00 USDT</div>

</div>

<div class="balance-info" id="short-balance">

<div class="data-title">空倉價值</div>

<div class="balance-value" id="short-value">0.00 USDT</div>

</div>

<div class="balance-info" id="long-margin">

<div class="data-title">可平多倉</div>

<div class="balance-value" id="long-margin-value">0.00 HRK</div>

</div>

<div class="balance-info" id="short-margin">

<div class="data-title">可平空倉</div>

<div class="balance-value" id="short-margin-value">0.00 HRK</div>

</div>

<div class="balance-info" id="spot-holding" style="display: none;">

<div class="data-title">現貨持有</div>

<div class="balance-value" id="spot-value">0.00 HRK</div>

</div>

<div class="balance-info" id="balance-info-default">

<div class="data-title">账户余额</div>

<div class="balance-value" id="balance-value">100.00 USDT</div>

</div>

<div class="balance-info" id="margin-info-default">

<div class="data-title">可用保证金</div>

<div class="balance-value" id="margin-value">100.00 USDT</div>

</div>

</div>

</div>

<div class="positions-section">

<h3>持仓</h3>

<table class="positions-table">

<thead>

<tr>

<th>币种</th>

<th>种类</th>

<th>方向</th>

<th>槓桿</th>

<th>数量</th>

<th>成本</th>

<th>最大持倉</th>

<th>开仓价</th>

<th>预估强平价</th>

<th>当前盈亏金额</th>

<th>当前盈亏%</th>

<th>买入时间</th>

<th>买入总耗時</th>

</tr>

</thead>

<tbody id="positions-body">

</tbody>

</table>

</div>

<div class="history-section">

<h3>历史成交</h3>

<table class="history-completed-table">

<thead>

<tr>

<th>币种</th>

<th>种类</th>

<th>方向</th>

<th>槓桿</th>

<th>数量</th>

<th>成本</th>

<th>最大持倉</th>

<th>开仓价</th>

<th>平仓价</th>

<th>最终盈亏金额</th>

<th>最终盈亏%</th>

<th>买入时间</th>

<th>卖出时间</th>

</tr>

</thead>

<tbody id="history-completed-body">

</tbody>

</table>

</div>

<div class="warning">

投资有风险，入市需谨慎

</div>

<footer>

<p>Heark 模拟交易平台 - 虚拟货币价格仅为演示用途，并非真实交易数据</p>

</footer>

</div>

<!-- 控制面板模态框 -->

<div class="modal" id="control-modal">

<div class="modal-content">

<div class="modal-header">

<h2>控制面板</h2>

<button class="close-modal" id="close-modal">&times;</button>

</div>

<div class="modal-controls">

<div class="modal-control">

<label>設置資產 (USDT):</label>

<input type="number" id="reset-balance" value="100" min="0">

<button class="modal-btn modal-btn-primary" id="reset-asset-btn">設置</button>

</div>

<div class="modal-control">

<label>时间比例 (秒/分钟):</label>

<input type="number" id="time-ratio" value="1" min="1" max="60">

</div>

<div class="modal-control">

<label>价格更新间隔 (秒):</label>

<input type="number" id="update-interval" value="1" min="0.1" step="0.1">

</div>

<div class="modal-control">

<label>上涨概率 (%):</label>

<input type="number" id="up-probability" value="50" min="0" max="100">

</div>

<div class="modal-control">

<label>波动率:</label>

<input type="number" id="volatility" value="0.1" min="0.1" max="10" step="0.1">

</div>

<div class="modal-control">

<label>自动交易:</label>

<input type="checkbox" id="auto-trade">

</div>

<div class="modal-control">

<label>调整价格 (USDT):</label>

<input type="number" id="adjust-price" value="" step="0.01">

<button class="modal-btn modal-btn-primary" id="set-price-btn">设置</button>

</div>

</div>

<div class="modal-buttons">

<button class="modal-btn pause-btn active" id="pause-btn">暫停中</button>

<button class="modal-btn modal-btn-primary" id="reset-chart">重置图表</button>

<button class="modal-btn modal-btn-primary" id="apply-settings">应用设置</button>

</div>

<div class="success-message" id="success-message">設置已成功應用</div>

<div class="success-message" id="pause-message">暫停已成功</div>

<div class="success-message" id="reset-message">圖表已重置並暫停</div>

<div class="success-message" id="price-message">價格已調整</div>

</div>

</div>

<script>

// 全局变量

let currentCoin = 'HRK';

let coins = {

HRK: { price: 5000, minuteHistory: [], change: 0, openPrice: 5000 },

NJC: { price: 1000, minuteHistory: [], change: 0, openPrice: 1000 }

};

let balance = 100;

let spotHoldings = {

HRK: 0,

NJC: 0

};

let positions = []; // 合约持仓

let spotPositions = []; // 现货持仓

let completedTrades = [];

let leverage = 1;

let timeframe = 15;

let upProbability = 50;

let volatility = 0.1;

let timeRatio = 1;

let updateInterval = 1;

let autoTrade = false;

let currentCandle = null;

let candleInterval = null;

let tradeHistory = [];

let chartData = [];

let chart;

let candleSeries;

let tradeMode = 'spot'; // 'spot' or 'contract'

let contractMode = 'open'; // 'open' or 'close'

let pageLoadTime = Math.floor(Date.now() / 1000); // 页面加载时间作为基准，取整到分钟

let isPaused = true; // 初始暫停

// 创建图表

function createChart() {

const chartContainer = document.getElementById('chart');

chart = LightweightCharts.createChart(chartContainer, {

width: chartContainer.clientWidth,

height: 420,

layout: {

backgroundColor: '#131722',

textColor: '#d9d9d9',

},

grid: {

vertLines: {

color: 'rgba(42, 46, 57, 0.5)',

},

horzLines: {

color: 'rgba(42, 46, 57, 0.5)',

},

},

timeScale: {

timeVisible: true,

secondsVisible: false,

barSpacing: 10,

},

rightPriceScale: {

borderColor: '#485c7b',

visible: true,

},

crosshair: {

mode: LightweightCharts.CrosshairMode.Normal,

},

});

candleSeries = chart.addCandlestickSeries({

upColor: '#00b15d',

downColor: '#ff5b5a',

borderDownColor: '#ff5b5a',

borderUpColor: '#00b15d',

wickDownColor: '#ff5b5a',

wickUpColor: '#00b15d',

});

}

// 生成分钟历史数据，从页面加载时间开始，初始为空

function generateMinuteHistory(coin) {

return []; // 初始为空，从当前时间开始构建

}

// 聚合K线数据，确保一致性和可见性

function aggregateKlines(minuteHistory, tf) {

if (minuteHistory.length === 0) return [];

const interval = tf \* 60;

const data = [];

let i = 0;

while (i < minuteHistory.length) {

const candleStart = Math.floor(minuteHistory[i].time / interval) \* interval;

const candles = [];

while (i < minuteHistory.length && Math.floor(minuteHistory[i].time / interval) \* interval === candleStart) {

candles.push(minuteHistory[i]);

i++;

}

if (candles.length > 0) {

const open = candles[0].open;

const close = candles[candles.length - 1].close;

const high = Math.max(...candles.map(c => c.high));

const low = Math.min(...candles.map(c => c.low));

data.push({

time: candleStart,

open: open,

high: high,

low: low,

close: close,

});

}

}

return data.slice(-200); // 限制但足够长时线可见

}

// 中心定位函数 - 改进：让最新K线在时间轴和价格轴的中心

function centerChart() {

if (chartData.length === 0) return;

const lastIndex = chartData.length - 1;

const lastData = chartData[lastIndex];

// 时间轴：假设显示50个bar，让最新bar在中间（从左到右第25个）

const barCount = 50;

const centerIndex = lastIndex - Math.floor(barCount / 2);

const fromIndex = Math.max(0, centerIndex);

chart.timeScale().setVisibleLogicalRange({ from: fromIndex, to: fromIndex + barCount });

// 价格轴：让当前价格在垂直中心

const priceRange = candleSeries.priceScale().getVisiblePriceRange();

if (priceRange) {

const centerPrice = lastData.close;

const rangeSize = priceRange.to - priceRange.from;

const newFrom = centerPrice - rangeSize / 2;

const newTo = centerPrice + rangeSize / 2;

candleSeries.priceScale().setVisiblePriceRange({ from: newFrom, to: newTo });

}

}

// 初始化当前蜡烛

function initCurrentCandle() {

const now = Math.floor(Date.now() / 1000);

const candleTime = Math.floor(now / (timeframe \* 60)) \* (timeframe \* 60);

const coin = coins[currentCoin];

const lastClose = chartData.length > 0 ? chartData[chartData.length - 1].close : coin.openPrice;

currentCandle = {

time: candleTime,

open: lastClose,

high: coin.price,

low: coin.price,

close: coin.price

};

if (chartData.length === 0 || chartData[chartData.length - 1].time !== candleTime) {

chartData.push(currentCandle);

}

candleSeries.setData(chartData);

// 中心定位

setTimeout(centerChart, 100); // 延迟以确保图表渲染

}

// 更新K线

function updateCandle() {

const coin = coins[currentCoin];

const now = Math.floor(Date.now() / 1000);

const candleTime = Math.floor(now / (timeframe \* 60)) \* (timeframe \* 60);

let updated = false;

if (currentCandle && currentCandle.time === candleTime) {

currentCandle.close = coin.price;

currentCandle.high = Math.max(currentCandle.high, coin.price);

currentCandle.low = Math.min(currentCandle.low, coin.price);

updated = true;

}

if (!updated) {

const open = chartData.length > 0 ? chartData[chartData.length - 1].close : coin.openPrice;

currentCandle = {

time: candleTime,

open: open,

high: coin.price,

low: coin.price,

close: coin.price

};

chartData.push(currentCandle);

}

candleSeries.setData(chartData);

}

// 更新价格函数，同时更新minuteHistory的最后一个蜡烛

function updatePrice() {

if (isPaused) return; // 如果暫停則不更新

const coin = coins[currentCoin];

const nowSec = Math.floor(Date.now() / 1000);

const currentMin = Math.floor(nowSec / 60) \* 60;

const isUp = Math.random() \* 100 < upProbability;

const changePercent = (Math.random() \* volatility) / 100;

const change = isUp ? coin.price \* changePercent : -coin.price \* changePercent;

coin.price = Math.max(0.01, coin.price + change);

coin.change = ((coin.price - coin.openPrice) / coin.openPrice \* 100);

// 更新或添加分钟历史，从加载时间开始

if (coins[currentCoin].minuteHistory.length === 0) {

coins[currentCoin].minuteHistory.push({

time: pageLoadTime,

open: coin.openPrice,

high: coin.openPrice,

low: coin.openPrice,

close: coin.openPrice

});

}

let lastCandle = coins[currentCoin].minuteHistory[coins[currentCoin].minuteHistory.length - 1];

if (lastCandle.time === currentMin) {

lastCandle.close = coin.price;

lastCandle.high = Math.max(lastCandle.high, coin.price);

lastCandle.low = Math.min(lastCandle.low, coin.price);

} else {

const newCandle = {

time: currentMin,

open: lastCandle.close,

high: coin.price,

low: coin.price,

close: coin.price

};

coins[currentCoin].minuteHistory.push(newCandle);

}

const priceEl = document.getElementById('current-price');

const changeEl = document.getElementById('price-change');

priceEl.textContent = `$${coin.price.toFixed(2)}`;

changeEl.textContent = `${coin.change >= 0 ? '+' : ''}${coin.change.toFixed(2)}%`;

const isUpChange = coin.change >= 0;

priceEl.className = `coin-price ${isUpChange ? 'price-up' : 'price-down'}`;

changeEl.className = `price-change ${isUpChange ? 'price-up' : 'price-down'}`;

// 更新左邊幣種價格

const currentCoinEl = document.querySelector('.coin-option.active');

const priceSmallEl = currentCoinEl.querySelector('.coin-price-small');

const changeSmallEl = currentCoinEl.querySelector('.coin-change-small');

priceSmallEl.textContent = `$${coin.price.toFixed(2)}`;

changeSmallEl.textContent = `${coin.change >= 0 ? '+' : ''}${coin.change.toFixed(2)}%`;

changeSmallEl.className = `coin-change-small ${isUpChange ? 'price-up-small' : 'price-down-small'}`;

// 重新聚合并更新图表

chartData = aggregateKlines(coins[currentCoin].minuteHistory, timeframe);

updateCandle();

candleSeries.setData(chartData);

updateOrderBook();

checkStops();

updatePositionsDisplay();

if (autoTrade && Math.random() < 0.05) {

const amount = parseFloat(document.getElementById('amount').value) || 0.01;

if (Math.random() < 0.5) {

openPosition(amount, 'long');

} else {

openPosition(amount, 'short');

}

}

}

// 生成订单簿数据

function generateOrderBook() {

const coin = coins[currentCoin];

const orderBook = { bids: [], asks: [] };

const spread = coin.price \* 0.001;

let bidPrice = coin.price \* 0.999;

for (let i = 0; i < 10; i++) {

const amount = (Math.random() \* 10 + 0.1).toFixed(4);

orderBook.bids.push({

price: bidPrice,

amount,

total: (bidPrice \* parseFloat(amount)).toFixed(2)

});

bidPrice -= spread \* 0.1;

}

orderBook.bids.sort((a, b) => b.price - a.price);

let askPrice = coin.price \* 1.001;

for (let i = 0; i < 10; i++) {

const amount = (Math.random() \* 10 + 0.1).toFixed(4);

orderBook.asks.push({

price: askPrice,

amount,

total: (askPrice \* parseFloat(amount)).toFixed(2)

});

askPrice += spread \* 0.1;

}

orderBook.asks.sort((a, b) => a.price - b.price);

return orderBook;

}

// 更新订单簿显示

function updateOrderBook() {

const orderBook = generateOrderBook();

const tbody = document.getElementById('order-book-body');

tbody.innerHTML = '';

orderBook.asks.slice().reverse().forEach(ask => {

const row = tbody.insertRow();

row.innerHTML = `<td class="ask-price">${ask.price.toFixed(2)}</td><td>${ask.amount}</td><td>${ask.total}</td>`;

});

const currentRow = tbody.insertRow();

currentRow.style.backgroundColor = '#2a3040';

currentRow.innerHTML = `<td style="color: #2a9df4; font-weight: bold;">${coins[currentCoin].price.toFixed(2)}</td><td>--</td><td>--</td>`;

orderBook.bids.forEach(bid => {

const row = tbody.insertRow();

row.innerHTML = `<td class="bid-price">${bid.price.toFixed(2)}</td><td>${bid.amount}</td><td>${bid.total}</td>`;

});

}

// 添加交易历史，隨機排列

function addTradeHistory(price, amount, direction) {

tradeHistory.unshift({ time: new Date(), price, amount, direction });

if (tradeHistory.length > 50) tradeHistory.pop();

// 隨機打亂

tradeHistory.sort(() => Math.random() - 0.5);

const tbody = document.getElementById('trade-history-body');

tbody.innerHTML = '';

tradeHistory.forEach(trade => {

const row = tbody.insertRow();

const timeStr = trade.time.toLocaleTimeString('zh-CN');

const dirText = trade.direction === 'buy' ? '买入' : '卖出';

const dirClass = trade.direction === 'buy' ? 'history-buy' : 'history-sell';

row.innerHTML = `<td>${timeStr}</td><td>${trade.price.toFixed(2)}</td><td>${trade.amount.toFixed(4)}</td><td class="${dirClass}">${dirText}</td>`;

});

}

// 更新持仓显示 - 改进：合并合约和现货持仓，方向调整，持续时间固定计算，强平只在负100%时

function updatePositionsDisplay() {

const tbody = document.getElementById('positions-body');

tbody.innerHTML = '';

// 合并持仓，按开仓时间降序排列，新开仓在上

const allPositions = [...positions, ...spotPositions].sort((a, b) => b.openTime - a.openTime);

allPositions.forEach(pos => {

const row = tbody.insertRow();

const currentPrice = coins[pos.coin].price;

let pnl, pnlPercent, liquidationPrice, leverageText, dirText, typeText, cost, maxPosition;

if (pos.type === 'spot') {

// 现货

pnl = (currentPrice - pos.entryPrice) \* pos.amount;

pnlPercent = (pnl / (pos.amount \* pos.entryPrice)) \* 100;

leverageText = '1x';

dirText = '買入';

typeText = '现货';

cost = (pos.amount \* pos.entryPrice).toFixed(2);

maxPosition = cost;

liquidationPrice = 'N/A';

} else {

// 合约

pnl = (currentPrice - pos.entryPrice) \* pos.amount \* (pos.direction === 'long' ? 1 : -1) \* pos.leverage;

pnlPercent = (pnl / (pos.amount \* pos.entryPrice)) \* 100;

leverageText = `${pos.leverage}x`;

dirText = pos.direction === 'long' ? '开多' : '开空';

typeText = '合约';

cost = (pos.amount \* pos.entryPrice).toFixed(2);

maxPosition = (parseFloat(cost) \* pos.leverage).toFixed(2);

liquidationPrice = pos.direction === 'long' ? pos.entryPrice \* (1 - 1 / pos.leverage) : pos.entryPrice \* (1 + 1 / pos.leverage);

liquidationPrice = liquidationPrice.toFixed(2);

}

// 只在亏损100%时强制平仓

if (pnlPercent <= -100) {

closePosition(pos.amount, pos.direction, pos.type === 'spot');

return;

}

const pnlClass = pnl >= 0 ? 'price-up' : 'price-down';

const openTimeStr = pos.openTime.toLocaleString('zh-CN');

const durationSec = ((Date.now() / 1000) - (pos.openTime.getTime() / 1000)).toFixed(1);

row.innerHTML = `<td>${pos.coin}/USDT</td><td>${typeText}</td><td>${dirText}</td><td>${leverageText}</td><td>${pos.amount.toFixed(4)}</td><td>${cost}</td><td>${maxPosition}</td><td>${pos.entryPrice.toFixed(2)}</td><td>${liquidationPrice}</td><td class="${pnlClass}">${pnl.toFixed(2)}</td><td class="${pnlClass}">${pnlPercent.toFixed(2)}%</td><td>${openTimeStr}</td><td>${durationSec}秒</td>`;

});

}

// 更新歷史成交 - 改进：现货方向为"卖出"

function updateCompletedTradesDisplay() {

const tbody = document.getElementById('history-completed-body');

tbody.innerHTML = '';

// 按平仓时间降序排列，新完成的在上

const sortedTrades = [...completedTrades].sort((a, b) => b.closeTime - a.closeTime);

sortedTrades.forEach(trade => {

const row = tbody.insertRow();

let dirText, leverageText, typeText, cost, maxPosition;

if (trade.type === 'spot') {

dirText = '賣出'; // 现货历史方向固定为卖出

leverageText = '1x';

typeText = '现货';

cost = (trade.amount \* trade.entryPrice).toFixed(2);

maxPosition = cost;

} else {

dirText = trade.direction === 'long' ? '开多' : '开空';

leverageText = `${trade.leverage}x`;

typeText = '合约';

cost = (trade.amount \* trade.entryPrice).toFixed(2);

maxPosition = (parseFloat(cost) \* trade.leverage).toFixed(2);

}

const pnlPercent = (trade.pnl / (trade.amount \* trade.entryPrice)) \* 100;

const pnlClass = trade.pnl >= 0 ? 'price-up' : 'price-down';

const openTimeStr = trade.openTime.toLocaleString('zh-CN');

const closeTimeStr = trade.closeTime.toLocaleString('zh-CN');

row.innerHTML = `<td>${trade.coin}/USDT</td><td>${typeText}</td><td>${dirText}</td><td>${leverageText}</td><td>${trade.amount.toFixed(4)}</td><td>${cost}</td><td>${maxPosition}</td><td>${trade.entryPrice.toFixed(2)}</td><td>${trade.exitPrice.toFixed(2)}</td><td class="${pnlClass}">${trade.pnl.toFixed(2)}</td><td class="${pnlClass}">${pnlPercent.toFixed(2)}%</td><td>${openTimeStr}</td><td>${closeTimeStr}</td>`;

});

}

// 开仓函数 - 改进：现货添加到spotPositions

function openPosition(amount, direction) {

const inputPrice = parseFloat(document.getElementById('price').value);

if (tradeMode === 'spot') {

const cost = amount \* inputPrice;

if (balance < cost) {

alert('余额不足！');

return;

}

balance -= cost;

spotHoldings[currentCoin] += amount;

spotPositions.push({ coin: currentCoin, amount, entryPrice: inputPrice, openTime: new Date(), type: 'spot' });

addTradeHistory(inputPrice, amount, 'buy');

} else {

const margin = (amount \* inputPrice) / leverage;

if (balance < margin) {

alert('保证金不足！');

return;

}

balance -= margin;

positions.push({ coin: currentCoin, amount, entryPrice: inputPrice, leverage, direction, openTime: new Date(), type: 'contract' });

addTradeHistory(inputPrice, amount, direction === 'long' ? 'buy' : 'sell');

}

updateBalanceDisplay();

updatePositionsDisplay();

updateCompletedTradesDisplay();

}

// 平仓函数 - 改进：现货从spotPositions移除并添加到历史

function closePosition(amount, direction, isSpot = false) {

let posArray = isSpot ? spotPositions : positions;

const posIndex = posArray.findIndex(p => p.coin === currentCoin && p.direction === direction && p.amount >= amount);

if (posIndex === -1) return;

const pos = posArray[posIndex];

const currentPrice = coins[currentCoin].price;

let pnl;

if (isSpot) {

pnl = (currentPrice - pos.entryPrice) \* amount;

balance += amount \* currentPrice;

spotHoldings[currentCoin] -= amount;

} else {

pnl = (currentPrice - pos.entryPrice) \* amount \* (direction === 'long' ? 1 : -1) \* pos.leverage;

balance += (amount \* pos.entryPrice / pos.leverage) + pnl;

}

completedTrades.push({

time: new Date(),

coin: currentCoin,

direction: direction,

entryPrice: pos.entryPrice,

exitPrice: currentPrice,

amount,

pnl,

leverage: pos.leverage || 1,

type: isSpot ? 'spot' : 'contract',

openTime: pos.openTime,

closeTime: new Date()

});

posArray.splice(posIndex, 1);

addTradeHistory(currentPrice, amount, direction === 'long' ? 'sell' : 'buy');

updateBalanceDisplay();

updatePositionsDisplay();

updateCompletedTradesDisplay();

}

// 执行交易 - 改进：现货买/卖逻辑调整

function executeTrade(amount, action) {

const inputPrice = parseFloat(document.getElementById('price').value);

if (tradeMode === 'spot') {

if (action === 'buy') {

// 买入添加到持仓

openPosition(amount, 'buy'); // 使用openPosition处理

} else {

// 卖出从持仓移除

closePosition(amount, 'buy', true); // direction for spot as 'buy' to match

}

} else { // 合约

const direction = contractMode === 'open' ? (action === 'buy' ? 'long' : 'short') : (action === 'buy' ? 'short' : 'long'); // 平多卖出平空买入

const isOpen = contractMode === 'open';

if (isOpen) {

openPosition(amount, direction);

} else {

closePosition(amount, direction, false);

}

}

updateBalanceDisplay();

updatePositionsDisplay();

updateCompletedTradesDisplay();

}

// 检查止损止盈

function checkStops() {

const currentP = coins[currentCoin].price;

const sl = parseFloat(document.getElementById('stop-loss').value);

const tp = parseFloat(document.getElementById('take-profit').value);

// 只检查合约持仓

positions.forEach((pos, index) => {

if (pos.coin !== currentCoin) return;

let triggerSl = false, triggerTp = false;

if (pos.direction === 'long') {

triggerSl = sl > 0 && currentP <= sl;

triggerTp = tp > 0 && currentP >= tp;

} else {

triggerSl = sl > 0 && currentP >= sl;

triggerTp = tp > 0 && currentP <= tp;

}

if (triggerSl || triggerTp) {

closePosition(pos.amount, pos.direction, false);

}

});

}

// 更新余额显示 - 改进：现货持有计算

function updateBalanceDisplay() {

// 默认显示

document.getElementById('balance-value').textContent = `${balance.toFixed(2)} USDT`;

// 现货持有显示

if (tradeMode === 'spot') {

const spotAmount = spotPositions.reduce((sum, pos) => sum + (pos.coin === currentCoin ? pos.amount : 0), 0);

document.getElementById('spot-value').textContent = `${spotAmount.toFixed(4)} ${currentCoin}`;

document.getElementById('spot-holding').style.display = 'block';

document.getElementById('margin-info-default').style.display = 'none';

} else {

document.getElementById('spot-holding').style.display = 'none';

document.getElementById('margin-value').textContent = `${(balance \* leverage).toFixed(2)} USDT`;

document.getElementById('margin-info-default').style.display = 'block';

}

// 平仓模式下显示持仓价值和可平仓量

if (tradeMode === 'contract' && contractMode === 'close') {

let longValue = 0, longAmount = 0;

let shortValue = 0, shortAmount = 0;

positions.forEach(pos => {

if (pos.coin === currentCoin) {

const value = pos.amount \* coins[currentCoin].price;

if (pos.direction === 'long') {

longValue += value;

longAmount += pos.amount;

} else {

shortValue += value;

shortAmount += pos.amount;

}

}

});

document.getElementById('long-value').textContent = `${longValue.toFixed(2)} USDT`;

document.getElementById('short-value').textContent = `${shortValue.toFixed(2)} USDT`;

document.getElementById('long-margin-value').textContent = `${longAmount.toFixed(4)} ${currentCoin}`;

document.getElementById('short-margin-value').textContent = `${shortAmount.toFixed(4)} ${currentCoin}`;

// 显示平仓相关，隐藏默认

document.getElementById('long-balance').style.display = 'block';

document.getElementById('short-balance').style.display = 'block';

document.getElementById('long-margin').style.display = 'block';

document.getElementById('short-margin').style.display = 'block';

document.getElementById('balance-info-default').style.display = 'none';

document.getElementById('margin-info-default').style.display = 'none';

} else {

// 隐藏平仓相关，显示默认

document.getElementById('long-balance').style.display = 'none';

document.getElementById('short-balance').style.display = 'none';

document.getElementById('long-margin').style.display = 'none';

document.getElementById('short-margin').style.display = 'none';

document.getElementById('balance-info-default').style.display = 'block';

}

}

// 更新止盈止损显示

function updateStopInputs() {

const stopLossGroup = document.querySelector('.input-group.stop-loss');

const takeProfitGroup = document.querySelector('.input-group.take-profit');

if (tradeMode === 'spot' || contractMode === 'open') {

stopLossGroup.classList.remove('hidden');

takeProfitGroup.classList.remove('hidden');

} else {

stopLossGroup.classList.add('hidden');

takeProfitGroup.classList.add('hidden');

}

}

// 更新买入/卖出按钮文本和功能

function updateActionButtons() {

const buyBtn = document.getElementById('buy-btn');

const sellBtn = document.getElementById('sell-btn');

if (tradeMode === 'spot') {

buyBtn.textContent = '买入';

sellBtn.textContent = '卖出';

} else if (contractMode === 'open') {

buyBtn.textContent = '开多';

sellBtn.textContent = '开空';

} else {

buyBtn.textContent = '平多';

sellBtn.textContent = '平空';

}

updateStopInputs();

updateBalanceDisplay();

}

// 初始化

function init() {

// 生成历史数据为空，从当前开始

Object.keys(coins).forEach(coin => {

coins[coin].minuteHistory = generateMinuteHistory(coin);

coins[coin].openPrice = coins[coin].price; // 以当前价格作为开盘

});

chartData = aggregateKlines(coins[currentCoin].minuteHistory, timeframe);

createChart();

candleSeries.setData(chartData);

initCurrentCandle();

// 初始暫停，不啟動間隔

updateOrderBook();

updateBalanceDisplay();

updatePositionsDisplay();

updateCompletedTradesDisplay();

updateActionButtons();

updatePrice();

// 复制市价

document.getElementById('copy-price').addEventListener('click', () => {

document.getElementById('price').value = coins[currentCoin].price.toFixed(2);

});

// 定位按鈕

document.getElementById('fit-view-btn').addEventListener('click', () => {

centerChart();

});

// 交易模式切换

document.querySelectorAll('.mode-option').forEach(option => {

option.addEventListener('click', (e) => {

e.preventDefault();

document.querySelectorAll('.mode-option').forEach(opt => opt.classList.remove('active'));

option.classList.add('active');

tradeMode = option.dataset.mode;

const leverageSelector = document.querySelector('.leverage-selector');

const contractButtons = document.querySelector('.contract-mode-buttons');

if (tradeMode === 'contract') {

leverageSelector.style.display = 'flex';

contractButtons.style.display = 'flex';

contractMode = 'open';

document.getElementById('open-position-btn').classList.add('active');

document.getElementById('close-position-btn').classList.remove('active');

} else {

leverageSelector.style.display = 'none';

contractButtons.style.display = 'none';

}

updateActionButtons();

});

});

// 合约开仓/平仓切换

document.getElementById('open-position-btn').addEventListener('click', (e) => {

e.preventDefault();

contractMode = 'open';

document.getElementById('open-position-btn').classList.add('active');

document.getElementById('close-position-btn').classList.remove('active');

updateActionButtons();

});

document.getElementById('close-position-btn').addEventListener('click', (e) => {

e.preventDefault();

contractMode = 'close';

document.getElementById('open-position-btn').classList.remove('active');

document.getElementById('close-position-btn').classList.add('active');

updateActionButtons();

});

// 杠杆

document.querySelectorAll('.leverage-option').forEach(option => {

option.addEventListener('click', (e) => {

e.preventDefault();

document.querySelectorAll('.leverage-option').forEach(opt => opt.classList.remove('active'));

option.classList.add('active');

leverage = parseInt(option.dataset.value);

updateBalanceDisplay();

});

});

// 买/卖按钮

function setupTradeButtons() {

document.getElementById('buy-btn').addEventListener('click', (e) => {

e.preventDefault();

const amount = parseFloat(document.getElementById('amount').value);

if (amount > 0) executeTrade(amount, 'buy');

});

document.getElementById('sell-btn').addEventListener('click', (e) => {

e.preventDefault();

const amount = parseFloat(document.getElementById('amount').value);

if (amount > 0) executeTrade(amount, 'sell');

});

}

setupTradeButtons(); // 初始设置

// 时间框架

document.querySelectorAll('.timeframe-btn').forEach(btn => {

btn.addEventListener('click', (e) => {

e.preventDefault();

document.querySelectorAll('.timeframe-btn').forEach(b => b.classList.remove('active'));

btn.classList.add('active');

timeframe = parseInt(btn.dataset.tf) || 1440;

chartData = aggregateKlines(coins[currentCoin].minuteHistory, timeframe);

candleSeries.setData(chartData);

initCurrentCandle();

});

});

// 币种切换 (不刷新历史)

document.querySelectorAll('.coin-option').forEach(option => {

option.addEventListener('click', (e) => {

e.preventDefault();

if (option.classList.contains('active')) return;

document.querySelectorAll('.coin-option').forEach(opt => opt.classList.remove('active'));

option.classList.add('active');

currentCoin = option.dataset.coin;

document.getElementById('current-coin').textContent = `${currentCoin}/USDT`;

document.getElementById('trade-pair').textContent = `${currentCoin}/USDT`;

chartData = aggregateKlines(coins[currentCoin].minuteHistory, timeframe);

candleSeries.setData(chartData);

initCurrentCandle();

updatePrice();

updateBalanceDisplay();

});

});

// 订单簿标签

document.getElementById('order-tab').addEventListener('click', (e) => {

e.preventDefault();

document.getElementById('order-tab').classList.add('active');

document.getElementById('trade-tab').classList.remove('active');

document.getElementById('order-table').style.display = 'table';

document.getElementById('trade-history').style.display = 'none';

});

document.getElementById('trade-tab').addEventListener('click', (e) => {

e.preventDefault();

document.getElementById('trade-tab').classList.add('active');

document.getElementById('order-tab').classList.remove('active');

document.getElementById('order-table').style.display = 'none';

document.getElementById('trade-history').style.display = 'block';

});

// 控制面板

document.getElementById('control-panel-btn').addEventListener('click', (e) => {

e.preventDefault();

const orderBook = document.querySelector('.order-book');

const modal = document.getElementById('control-modal');

modal.style.left = orderBook.offsetLeft + 'px';

modal.style.height = orderBook.getBoundingClientRect().bottom + 'px';

modal.style.display = 'flex';

});

document.getElementById('close-modal').addEventListener('click', (e) => {

e.preventDefault();

document.getElementById('control-modal').style.display = 'none';

document.getElementById('success-message').style.display = 'none';

document.getElementById('pause-message').style.display = 'none';

document.getElementById('reset-message').style.display = 'none';

document.getElementById('price-message').style.display = 'none';

});

window.addEventListener('click', (e) => {

if (e.target === document.getElementById('control-modal')) {

document.getElementById('control-modal').style.display = 'none';

document.getElementById('success-message').style.display = 'none';

document.getElementById('pause-message').style.display = 'none';

document.getElementById('reset-message').style.display = 'none';

document.getElementById('price-message').style.display = 'none';

}

});

document.getElementById('reset-asset-btn').addEventListener('click', (e) => {

e.preventDefault();

const newBalance = parseFloat(document.getElementById('reset-balance').value) || 100;

balance = newBalance;

positions = [];

spotPositions = [];

updateBalanceDisplay();

updatePositionsDisplay();

alert(`资产设置為 ${newBalance} USDT`);

});

// 暫停按鈕

document.getElementById('pause-btn').addEventListener('click', (e) => {

e.preventDefault();

isPaused = !isPaused;

const btn = document.getElementById('pause-btn');

const pauseMsg = document.getElementById('pause-message');

if (isPaused) {

btn.textContent = '暫停中';

btn.classList.add('active');

clearInterval(candleInterval);

pauseMsg.textContent = '暫停已成功';

} else {

btn.textContent = '運行中';

btn.classList.remove('active');

candleInterval = setInterval(updatePrice, updateInterval \* 1000);

pauseMsg.textContent = '運行已成功';

}

pauseMsg.style.display = 'block';

setTimeout(() => {

pauseMsg.style.display = 'none';

}, 3000);

});

document.getElementById('apply-settings').addEventListener('click', (e) => {

e.preventDefault();

timeRatio = parseInt(document.getElementById('time-ratio').value) || 1;

updateInterval = parseFloat(document.getElementById('update-interval').value) || 1;

upProbability = parseInt(document.getElementById('up-probability').value) || 50;

volatility = parseFloat(document.getElementById('volatility').value) || 0.1;

autoTrade = document.getElementById('auto-trade').checked;

clearInterval(candleInterval);

if (!isPaused) {

candleInterval = setInterval(updatePrice, updateInterval \* 1000);

}

// 顯示成功訊息

const successMsg = document.getElementById('success-message');

successMsg.style.display = 'block';

setTimeout(() => {

successMsg.style.display = 'none';

}, 3000);

// 不关闭模态

});

// 調整價格

document.getElementById('set-price-btn').addEventListener('click', (e) => {

e.preventDefault();

const newPrice = parseFloat(document.getElementById('adjust-price').value);

if (newPrice && newPrice > 0) {

coins[currentCoin].price = newPrice;

coins[currentCoin].change = ((newPrice - coins[currentCoin].openPrice) / coins[currentCoin].openPrice \* 100);

updatePrice();

const priceMsg = document.getElementById('price-message');

priceMsg.style.display = 'block';

setTimeout(() => {

priceMsg.style.display = 'none';

}, 3000);

}

});

document.getElementById('reset-chart').addEventListener('click', (e) => {

e.preventDefault();

// 暫停

isPaused = true;

document.getElementById('pause-btn').textContent = '暫停中';

document.getElementById('pause-btn').classList.add('active');

clearInterval(candleInterval);

Object.keys(coins).forEach(coin => {

coins[coin].minuteHistory = generateMinuteHistory(coin);

coins[coin].openPrice = coins[coin].price;

});

chartData = aggregateKlines(coins[currentCoin].minuteHistory, timeframe);

candleSeries.setData(chartData);

initCurrentCandle();

tradeHistory = [];

document.getElementById('trade-history-body').innerHTML = '';

const resetMsg = document.getElementById('reset-message');

resetMsg.style.display = 'block';

setTimeout(() => {

resetMsg.style.display = 'none';

}, 3000);

});

}

window.addEventListener('load', init);

window.addEventListener('resize', () => {

if (chart) {

chart.applyOptions({ width: document.getElementById('chart').clientWidth });

}

});

</script>

</body>

</html>