# TradingView Alerts Extractor

## Overview

Automated Node.js script using Puppeteer to extract TradingView alerts data and export to CSV format.

> **Type of Project:** Web Scraping
> **Technology Stack:** Node.js, Puppeteer, XLSX
> **Output Format:** CSV file
> **Browser Mode:** Non-headless (visible)

## Prerequisites

### Required Packages

```
npm install puppeteer
npm install xlsx
```

### Required Information

| Parameter | Description | Where to Find |
|---|---|---|
| YOURCHARTID | TradingView chart identifier | URL of your saved chart |
| YOURSESSIONID | TradingView session cookie | Browser DevTools → Application → Cookies |

> **IMPORTANT:** Replace YOURCHARTID and YOURSESSIONID with your actual values before running the script.

## Configuration

### Browser Settings

```
const browser = await puppeteer.launch({
    headless: false,        // Shows browser window
    defaultViewport: null,  // Uses default viewport
    args: ['--start-maximized']  // Starts maximized
});
```

### Authentication

```
await page.setCookie({
    name: 'sessionid',
    value: 'YOURSESSIONID',  // Replace with actual session ID
    domain: '.tradingview.com'
});
```

## Script Flow

### Execution Steps

1. **Browser Launch:** Opens Puppeteer browser in visible mode
2. **Navigation:** Goes to TradingView chart URL
3. **Authentication:** Sets session cookie and reloads page
4. **Panel Check:** Verifies if alerts panel is open, opens if needed
5. **List Navigation:** Clicks on list tab to show alerts
6. **Data Extraction:** Iterates through alerts, hovers for tooltips
7. **Scrolling:** Auto-scrolls to load more alerts
8. **CSV Generation:** Converts extracted data to CSV format
9. **Download:** Triggers browser download of CSV file

## Data Extraction Process

### Alert Processing Loop

```
while (true) {
    const alerts = await page.$$(alertSelector);  // Get visible alerts

    // Process each alert
    for (let i = 0; i < alerts.length; i++) {
        await alerts[i].hover();  // Hover to show tooltip
        await new Promise(r => setTimeout(r, 1500));  // Wait for tooltip

        // Extract tooltip text
        const tooltipText = await page.evaluate(() => {
            // Find element containing Strategy text
        });
    }

    // Scroll for more alerts
    await page.evaluate((container) => {
        el.scrollBy(0, 200);  // Incremental scroll
    });
}
```

### Data Parsing

| Field | Extraction Method |
|---|---|
| Alert Name | Line before "Strategy" text |
| Strategy Name | Text before opening parenthesis |
| Parameters | Content inside parentheses, split by comma |
| Symbol | First part of line after strategy |
| Timeframe | Parts separated by bullet (•) |
| Created Date | Line starting with "Created:" |

## Selectors Reference

### Key DOM Selectors

| Element | Purpose |
|---|---|
| Alerts Panel | Check if alerts widget is open |
| List Tab (#list) | Switch to list view of alerts |
| Alert Items | Individual alert entries |
| Scroll Container | Container for scrolling through alerts |

## Timing Configuration

### Wait Times

| Operation | Wait Time | Purpose |
|---|---|---|
| Page Load | 5000ms | Initial page load after cookie set |
| Panel Open | 2000ms | Wait for alerts panel to open |
| Tooltip Hover | 1500ms | Wait for tooltip to appear |
| Scroll | 1000ms | Wait after scrolling |
| Before Close | 3000ms | Wait before closing browser |

## Output Format

### CSV Structure

The generated CSV file contains extracted alert data in columnar format:

- Alert Name
- Strategy Name
- Strategy Parameters (multiple columns)
- Symbol
- Timeframe Components
- Creation Date

### File Output

```
Filename: tradingview_alerts.csv
Location: Browser default download folder
```

## Error Handling

### Scroll Detection

```
// Check if we've reached the end
const scrolledBefore = await page.evaluate((container) => {
    return el ? el.scrollTop : 0;
});

// Attempt to scroll
await page.evaluate((container) => {
    el.scrollBy(0, 200);
});

// Check new position
const scrolledAfter = await page.evaluate((container) => {
    return el ? el.scrollTop : 0;
});

// Exit if no more scrolling possible
if (scrolledBefore === scrolledAfter) {
    console.log('Reached end of list');
    break;
}
```

## Usage Instructions

1. **Install Dependencies:**

```
npm install puppeteer xlsx
```

2. **Get Session ID:**
   - Open TradingView in browser
   - Login to your account
   - Open DevTools (F12)
   - Go to Application → Cookies
   - Find and copy sessionid value
3. **Get Chart ID:**
   - Open your saved chart
   - Copy the ID from URL after /chart/
4. **Update Script:**
   - Replace YOURCHARTID with actual chart ID
   - Replace YOURSESSIONID with actual session ID
5. **Run Script:**

```
node scriptname.js
```

## Console Output

### Log Messages

| Message | Meaning |
| --- | --- |
| Script started! | Script initialization |
| Starting browser... | Puppeteer launch |
| Going to chart... | Navigation started |
| Processing alert X... | Processing specific alert number |
| Reached end of list | All alerts processed |
| CSV download started | File download initiated |